Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

Т. Л. Романькова

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименной дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения

Гомель 2014

УДК 004.43(075.8)
ББК 32.973-018.2я73
Р69

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 4 от 25.11.2013 г.*)

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. физ.-мат. наук *О. А. Кравченко*

Романькова, Т. Л.

Р69 Объектно-ориентированное программирование : лаборатор. практикум по одноим. дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / Т. Л. Романькова. – Гомель : ГГТУ им. П. О. Сухого, 2014. – 84 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://library.gstu.by. – Загл. с титул. экрана.

Лабораторный практикум содержит набор вариантов заданий для изучения современного языка программирования C#.

Для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения ИПК и ПК

УДК 004.43(075.8)
ББК 32.973-018.2я73

СОДЕРЖАНИЕ

1. Создание простого консольного приложение в среде Microsoft Visual C# Express Edition

Цель: *освоить средства **Microsoft Visual C# Express Edition** для создания и отладки консольного приложения.*

1.1. Краткие теоретические сведения

Все переменные, используемые в программе должны быть описаны явным способом.

При описании задаются ее имя и тип:

тип имя1,[имя2, имя3, …];

Например,

int dlina; float b;

При описании можно инициализировать переменные:

int x = 1, b;

float y = 0.1f, n = y * 2 + 2.1;

Некоторые числовые типы данных приведены в таблице 1.1.

Таблица 1.1

имя	Тип CTS	Описание	Диапазон
int	System.Int32	32-битное целое значение	-2147483648:2147483647
long	System.Int64	64-битное целое значение	-9223372036854775808: 9223372036854775807
byte	System.Byte	8-битное целое без знака	0:255
float	System.Single	32-битное с плавающей точкой одинарной точности	$\pm 1.5\times10^{-45}$**:** $\pm 3.4\times10^{38}$
double	System.Double	32-битное с плавающей точкой двойной точности	$\pm 5.0\times10^{-324}$**:** $\pm 1.7\times10^{308}$

В таблице 1.2 приводятся основные математические функции, реализованные в методах класса Math.

Таблица 1.2

Функция в математике	**Метод класса Math**
$\|x\|$	Abs(x)
Arccos x	Acos(x)
Arcsin x	Asin(x)
Arctan x	Atan(x)
Натуральный логарифм	Log(x)
Десятичный логарифм	Log10(x)
число *e*	E
число π	PI
e^x	Exp(x)
tg x	Tan(x)
x^y	Pow(x,y)
$\sqrt{x}$	Sqrt(x)

Для доступа к методу используется операция доступа (точка):

Math. Abs(x+2).

Для ввода данных можно использовать метод **ReadLine()** из класса Console. Этот метод возвращает строку символов.

Ввод чисел с клавиатуры можно осуществить в два этапа:

- ввести число в виде строки символов
- преобразовать строку в переменную соответствующего типа.

Преобразование выполняется с помощью класса **Convert** из пространства имен System, используя соответствующие методы класса **ToInt32(s)**, **ToDouble(s)**, **ToByte(s)** и т.д.

Здесь s – параметр типа string.

Например:

```
string s;
s = Console.ReadLine();
byte h = Convert.ToByte(s);
```

Преобразование можно также выполнить с помощью метода **Parse**, который есть в каждом стандартном арифметическом классе.

Например:

```
byte w = byte.Parse(s);
```

В классе Console существует несколько вариантов методов с именами **Write** и **WriteLine**, предназначенных для вывода значений различных типов.

Если метод **WriteLine (Write)** вызван с одним параметром, он может быть любого встроенного типа. Если требуется вывести в одной строке несколько величин различного типа, перед передачей для вывода их нужно «склеить» в одну строку с помощью операции +. Пример.

```
int  x = 3;
Console.WriteLine("x="+x);
```

Синтаксис оператора if:

```
if(выражение_1) оператор_1;
else if(выражение_2) оператор_2;
...
else if(выражение_K) оператор_K;
else оператор_N;
```

Формат оператора цикла с предусловием ***while***:

```
while ( выражение ) оператор;
```

Синтаксис метода:

```
[ атрибуты ] [ спецификаторы ] тип  имя   ([параметры])
 {
   тело метода
 }
```

Спецификаторы метода:

public	доступ не ограничен
protected	доступ только из данного и производных классов
internal	доступ только из данной сборки
private	доступ только из данного класса (по умолчанию)
static	статический метод

Тип в заголовке метода определяет тип результата работы метода.

Для передачи значения выражения в качестве результата работы метода используется оператор

```
return  выражение;
```

Если метод не возвращает никакого значения, в заголовке указывается тип **void,** а оператор **return** отсутствует.

Обращение к статическому методу класса:

имя класса. имя метода([аргументы])

Обращение к нестатическому методу класса:

имя объекта. имя метода([аргументы])

При вызове метода с параметрами количество аргументов должно совпадать с количеством параметров в заголовке метода. Кроме того, должно существовать неявное преобразование типа аргумента к типу соответствующего параметра.

1.2. Методические указания к выполнению работы

Для выполнения заданий из п.1.3.1- п.1.3.4 необходимо сначала разработать алгоритм решения задачи.

Затем создать консольное приложение, выполнив команду меню

File – New Project.

При создании консольного приложения средой автоматически создается следующая заготовка:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
      }
   }
}
```

Для решения задач из п.1.3.1- п.1.3.3 необходимо реализовать разработанный алгоритм в методе **Main**, применяя соответствующие операторы языка программирования.

В конце метода Main для остановки выполнения программы необходимо использовать оператор

Console.ReadKey( );

Сохранять созданную программу необходимо в отдельной папке командой **Save All**.

Пример.

Вычислить значение выражения $b = x(arctgZ - e^{-(x+3)})$.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
class Program
   {
      static void Main(string[] args)
      {
         float x,z;
         string s;
         Console.WriteLine("Введите x");
         s = Console.ReadLine();
         x = Convert.ToSingle(s);
         Console.WriteLine("Введите z");
         s = Console.ReadLine();
         z = Convert.ToSingle(s);
        double b = x * (Math.Atan(z) - Math.Exp(-(x + 3)));
         Console.WriteLine("x="+x + " z=" + z);
         Console.WriteLine("z=" + z);
         Console.ReadKey( );
      }
   }
}
```

Для решения задачи из п.1.3.4 необходимо сначала в классе Program создать два статических метода: для ввода значения переменной и для вычисления заданной функции. Затем следует использовать созданные методы в реализации алгоритма решения задачи в методе Main.

1.3. Варианты заданий

1.3.1. Программирование линейных алгоритмов

Разработать консольное приложение для вычисления значения функции, заданной в таблице 1.3.

Предусмотреть ввод исходных данных с клавиатуры и вывод на экран исходных данных и результата вычислений.

Таблица 1.3

Вариант	Вид функции
1	$f = \frac{x + a^2}{1{,}4a} - x$
2	$z = \frac{x^3 + b}{1{,}5 + b} - xb$
3	$b = \frac{y^2}{z + y^3 z} + 1{,}35$
4	$y = z^2 x^2 + \frac{1{,}2}{x + 2}$
5	$p = 1{,}834 + \frac{2{,}3 \cdot y - b^4}{b^2 + 1}$
6	$b = z(y - (x + y)^{-2})$
7	$f = -\frac{a}{2b} + \frac{1{,}3 + x}{a + b^3}$
8	$z = x^3 - \frac{1{,}7y}{x + y} + f^2$
9	$b = x\left(t + \frac{2{,}3 - x}{t^2}\right)$
10	$y = 1{,}6(z + 1)^2 + \frac{2 + x}{2{,}78}$
11	$p = (t^2 z + 2x) \cdot \frac{4{,}05 + t}{x^3}$
12	$f = 5z - 4t^2 + \frac{3{,}2 - tz}{3}$
13	$b = (z - x)\frac{y - z^3}{1 + (y - x)^2}$
14	$p = y^4 + \frac{3{,}23f + t}{3}y$
15	$y = \frac{3x - b}{a^3} - \frac{(a + b)^2}{2{,}5}$

1.3.2. Программирование разветвляющихся алгоритмов

Разработать консольное приложение для вычисления значения функции, заданной в таблице 1.4.

Предусмотреть ввод исходных данных с клавиатуры и вывод на экран исходных данных, вычисленного значения функции и номера формулы, по которой производился расчет.

Таблица 1.4

Вариант	Вид функции
1	$f=\begin{cases} x^2+z & \text{åñëè } -5<z\le 0 \\ 2{,}5z & \text{åñëè } z\le -5 \\ \dfrac{x^3+1{,}3}{z} & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
2	$f=\begin{cases} 2x+z & \text{åñëè } x\le 0 \\ 2{,}5x^2 & \text{åñëè } 0<x\le 3 \\ \dfrac{x^3}{10z} & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
3	$f=\begin{cases} 2x+\dfrac{z^3}{2{,}5} & \text{åñëè } -2\le z\le 2 \\ 3z^2 & \text{åñëè } 2<z \\ 2{,}3z-x & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
4	$f=\begin{cases} z^2x^2+\dfrac{1{,}2}{x+2}, & \text{åñëè } -5\le x<-2 \\ 3x+z, & \text{åñëè } x\ge -2 \\ 0.01x^4 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
5	$f=\begin{cases} \dfrac{2{,}3\cdot z-x^4}{z^2+1}, & \text{åñëè } -2\le z\le 2 \\ 2x+z, & \text{åñëè } z>2 \\ \dfrac{z^4}{20{,}3} & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
6	$f=\begin{cases} 2{,}5z-0{,}11x^2, & \text{åñëè } 3\le x \\ z(x-z)^2, & \text{åñëè } x<-3 \\ 2x^3 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
7	$f=\begin{cases} \dfrac{1{,}3+z}{x+z^3}, & \text{åñëè } z\ge 0 \\ 3(z+x)^2, & \text{åñëè } -6<z<-3 \\ -\dfrac{z}{2x} & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$

Вариант	Вид функции
8	$f=\begin{cases} x^3-\dfrac{1,2}{x+y}, & \text{åñëè } x<0 \\ z-x^2, & \text{åñëè } 0\le x<5 \\ x+\dfrac{z}{2,34} & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
9	$f=\begin{cases} 0.001z^3, & \text{åñëè } -10<z\le 0 \\ z+\dfrac{8,1-x}{z^2}, & \text{åñëè } 0<z<5 \\ 2,5z+x^2 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
10	$f=\begin{cases} \dfrac{z+x}{2,75}, & \text{åñëè } x\le 0 \\ 1,2(x+1)^2, & \text{åñëè } 0<x<3 \\ \dfrac{z}{1,3+z}x^2 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
11	$f=\begin{cases} \dfrac{2,05+z}{x^3}, & \text{åñëè } z\ge 0 \\ \dfrac{0.01z^3-2}{x+z}, & \text{åñëè } -10<z<-3 \\ x^2z^2+3,08 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
12	$f=\begin{cases} x^3+\dfrac{z^2}{5-x}, & \text{åñëè } 1\le x<5 \\ 2x^2-zx, & \text{åñëè } -3<x<1 \\ x+1,05 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
13	$f=\begin{cases} \dfrac{z^2}{5}+x, & \text{åñëè } 3\le z<6 \\ \dfrac{x-z^3}{1+(z-x)^2}, & \text{åñëè } 0<z<2 \\ 3z-3,05 & \text{â îñòàëüíûõ ñëó÷àÿõ} \end{cases}$
14	$f=\begin{cases} \left(\dfrac{z^2}{2,3}+x\right)^2, & \text{если } 0\le x<3 \\ x^3-2,2, & \text{если } x\le 0 \\ 2x+z^4 & \text{в остальных случаях} \end{cases}$

Вариант	Вид функции
15	$f=\begin{cases}\dfrac{3x-5}{z^3}, & \textit{если} \quad 0<z\le 2\\ \dfrac{(x+z)^2}{2{,}5}, & \textit{если} \quad 2<z<4\\ x^2z+1{,}03 & \textit{в остальных случаях}\end{cases}$

1.3.3. Программирование циклических алгоритмов

Разработать консольное приложение для вычисления значения функции, заданной в таблице 1.5, для аргумента x, принимающего значения в пределах от $x_{нач}$ до $x_{кон}$ с шагом Δx.

Результат выводить в виде таблицы.

Таблица 1.5

Вариант	Вид функции
1	$b=\dfrac{\sin x+3{,}7}{\lvert\cos x^3-2\rvert}$
2	$b=\dfrac{x^3}{x+3\cos^2 x^3}$
3	$b=\dfrac{\lg x+\cos\lvert x\rvert}{\cos^2 x^3}$
4	$b=\dfrac{\sqrt[3]{\lvert 2+x^2\rvert}\cdot e^{\sin x}}{x^2+0{,}5}$
5	$b=\sqrt{\lvert x\rvert+3}+\dfrac{\sin^2 x}{1{,}3}$
6	$b=\left(\sin^2 x+1\right)-\dfrac{\sqrt{\lvert x-3\rvert}}{3{,}01}$
7	$b=2{,}1+\dfrac{tg\,x-e^{-(x+3)}}{1{,}05}$
8	$b=x\left(3+\dfrac{\cos^2 x-3{,}5}{2}\right)$
9	$b=\dfrac{\sqrt{x+\sqrt[3]{x}}}{2}-3{,}2$
10	$b=\cos^3 x+\dfrac{2x+1{,}09}{\sqrt{\lvert x-2\rvert}}$
11	$b=e^{\lvert x-4\rvert}\left(\sin^2 x+1\right)^x$

12	$b = 0{,}2\sin x - e^x$
13	$b = \sqrt[3]{\lvert 2 - x\rvert}\,\dfrac{x - 3{,}1}{1 + (5 - x)^2}$
14	$b = \dfrac{\sin\left(\sqrt{x} + x^{2{,}5} + 3\right)}{\cos^2 x^2}$
15	$b = \sqrt{\dfrac{\lvert x\rvert + 1}{3{,}05 + x^2}}$

1.3.4. Разработка и использование методов класса

Составить программу для вычисления функции ***b=f(x,y,z)***, заданной в таблице 1.6, добавив в класс ***Program*** статические методы с параметрами для ввода исходных данных и вычисления функции. Предусмотреть вычисление функции не менее двух раз с различными аргументами.

Таблица 1.6

Вариант	Вид функции
1	$b = \dfrac{\sin x + \cos^2(x + z)}{\lvert\cos x^3 - 2y^2\rvert}$
2	$b = \dfrac{\lg x + \cos\lvert x + 2z\rvert}{\cos^2 x^3} - 1{,}7$
3	$b = \dfrac{z^3}{x + z\cos^2 y^3}$
4	$b = \sqrt{\lvert x\rvert + \sqrt[4]{\lvert y\rvert}} + \dfrac{\sin^2 z}{3{,}5}$
5	$b = \dfrac{\sqrt[3]{\lvert 2 + x^2\rvert}\cdot e^{\sin x} - \cos y}{z^2 + 2{,}5}$
6	$b = z + \dfrac{tg\ y - e^{-(x+3)}}{3{,}65}$
7	$b = \left(\sin^2 z + tgz\right) - \dfrac{\sqrt{\lvert x - y\rvert}}{2{,}1}$
8	$b = \dfrac{\sqrt{y + \sqrt[3]{x}}}{2z} - 1$
9	$b = x\left(tg\ z + \dfrac{\cos^2 y - 2{,}5}{z^3}\right)$

Вариант	Вид функции
10	$b = e^{\lvert x-y\rvert}\left(\ln^2 z + 1\right)^x$
11	$b = \cos^y z + \dfrac{tg\ 2x + \lvert y\rvert}{\sqrt{\lvert x-z\rvert}}$
12	$b = 5\sin z - e^{0,1y^2} + \lvert xy\rvert$
13	$b = \sqrt[3]{\lvert z-x\rvert}\,\dfrac{y-\ln z}{1+(y-x)^2}$
14	$b = y^z + \sqrt{\dfrac{\lvert x\rvert + \lvert y\rvert}{2,5+z^2}}$
15	$b = \dfrac{\sin\left(\sqrt{x} + y^{2,5} + 3\right)}{\lvert 2z\rvert - \cos^2 x^2}$

1.4. Контрольные вопросы

1. Основные понятия и принципы объектно-ориентированного программирования.
2. Классификация типов данных в C#.
3. Встроенные типы данных в C#.
4. Структура консольного приложения в C#.
5. Порядок создания и отладки простого консольного приложение в среде Microsoft Visual C# Express Edition
6. Вывод данных с помощью методов класса **System.Console**. Форматный вывод. Примеры.
7. Ввод данных с помощью методов класса **System.Console**. Примеры.
8. Основные математические функции, реализованные в методах класса Math. Примеры.
9. Синтаксис метода класса. Спецификаторы методов. Пример метода. Обращение к статическим и нестатическим методам.

2. Создание и использование классов

Цель: *Получить навыки использования методов класса Math, научиться создавать классы, содержащие поля и методы, конструкторы, свойства, научиться создавать и использовать объекты класса.*

2.1. Краткие теоретические сведения

Класс описывается следующим образом:

[спецификаторы] class Имя_Класса [: предки]
{
тело класса
}

Спецификаторы определяют свойства класса и доступность для других элементов программы. Допустимо использование следующих спецификаторов:

public - доступ не ограничен;

protected - для вложенных классов доступ только из элементов данного и производных классов;

internal - доступ только из данной программы;

private - для вложенных классов доступ только из элементов класса, внутри которого описан данный класс.

Для не вложенных классов используются только спецификаторы public и internal.

Объекты (экземпляры) класса создаются явным или неявным образом (программистом или системой). Для явного создания экземпляра используется операция **new**.

Формат операции: **new** Имя_класса ([аргументы]).

Например, пусть имеется следующее описание класса:

```
class Primer1{}
```

Тогда для создания объектов p1 и p2 в программе следует набрать

```
Primer1 p1 = new Primer1();
Primer1 p2 = new Primer1();
```

Переменные, описанные в классе, называются ***полями*** класса.

Синтаксис описания элемента данных:

[*спецификаторы*] [const] *тип имя* [= *начальное_значение*];

Обращение к статическому полю класса:

Имя_класса. имя_поля

Обращение к константе класса:

Имя_класса. имя_константы

Обращение к обычному полю (полю экземпляра) класса:

Имя_объекта. имя_поля

Конструктор объекта класса – это метод, предназначенный для инициализации объекта, автоматически вызываемый при создании объекта с помощью операции **new**. Имя конструктора должно совпадать с именем класса.

Описание:

[*спецификатор*] *Имя_класса*([*параметры*])
{ *тело конструктора*}

Обычно используется спецификатор public.

Конструктор не возвращает значение.

Для инициализации статических данных класса можно создать статический конструктор.

Статический конструктор вызывается автоматически до вызова конструктора объекта. Он должен быть закрытым.

Описание статического конструктора:

static *Имя_класса*( )
{ *тело конструктора*}

Свойство – это элемент класса, предоставляющий доступ к его полям. Обычно связано с закрытым полем класса.

[*спецификаторы*] *тип имя_свойства*
{
[get *код аксессора чтения поля*]
[set *код аксессора записи поля*]
}

Оба аксессора **не могут отсутствовать**.

При обращении к свойству автоматически вызываются аксессоры чтения и установки.

Обращение к свойству:

имя_объекта.имя_свойства

2.2. Методические указания к выполнению работы

При выполнении задания 2.3.1 после создания консольного приложения нужно создать класс, описывающий таблицу. Например,

```
class Table
   {
public static void TableHead
          (string head,string NameArg, int n1, string NameFun, int n2)
     {
Console.Write("╔");
      for (int i = 0; i < n1; i++) Console.Write("═");
Console.Write("╦");
for (int i = 0; i < n2; i++) Console.Write("═");
 Console.WriteLine("╗");
string s = "║{0,-" + n1.ToString( ) + "}║{1,-"+n2.ToString( ) +
"}║";
Console.WriteLine(s,NameArg , NameFun);
Console.Write("╠");
      for (int i = 0; i < n1; i++)   Console.Write("═");
Console.Write("╬");
      for (int i = 0; i < n2; i++) Console.Write("═");
      Console.WriteLine("╣");
     } // конец метода TableHead

public static void TableDown(int  n1,int   n2)
        {
          Console.Write("╚");
          for (int i = 0; i < n1; i++) Console.Write("═");
          Console.Write("╩");
          for (int i = 0; i < n2; i++) Console.Write("═");
          Console.WriteLine("╝");
        }
public static void TableLine(double x,int   n1,double   y, int   n2)
       {
          string  s = "║{0," + n1.ToString( ) + "}║{1," + n2.ToString()
+ ":f3}║";
          Console.WriteLine(s, x, y);
}
} //конец класса Table
```

В данном примере методы статические, а при выполнении своей работы нужно методы сделать нестатическими, а их параметры преобразовать в поля объектов класса. Также в классе необходимо предусмотреть конструктор, инициализирующий поля объекта заданными значениями.

Следующий этап – разработка класса для описания функции заданного вида. Например,

```
class Function
   {
     public  void  SetYZ(double  y, double  z)
          { this.y = y;    this.z = z; }
public  double  Mean(double  x)
{ return    (y*y+z*Math.Cos(x))/2.5;}
public  void  FunctionTable(double  xn, double  xk, double  dx)
{  Table.TableHead(ToString(), "x", 7, "f", 10);
   double   x = xn;
  while ((xn < xk) ? (x <= xk+dx/2) : (x >= xk-dx/2))
      {
        Table.TableLine(x, 7, Mean(x),10);
        x=(xn<xk)?(x+dx):(x-dx);
      }
  Table.TableDown(7, 10);
}

public   string   ToString()
{ string  s;
  s  =  "f(x)=("  +  (y  *  y).ToString()  +  "+"  +  z.ToString()  +
"*cos(x))/2.5";
return s;
}
public   static   void   InputXnXkDx
                  (out   double   xn, out   double   xk, out double
dx)
{
 Console.WriteLine("Введите начальное значение аргумента");
          xn = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("Введите конечное значение аргумента");
          xk = Convert.ToDouble(Console.ReadLine());
dx = 0;
```

```
bool   f = true;
while (f)
   { Console.WriteLine("Введите шаг изменения аргумента");
     dx = Convert.ToDouble(Console.ReadLine());
     if  (dx > 0)   f = false;
else
   Console.WriteLine("Шаг отрицательный или нулевой! Повторите ввод");
}      }    }
```

После разработки указанных в задании классов в методе Main класса Program нужно создать требуемое количество объектов указанных классов и вызвать от имени этих объектов элементы класса, необходимые для выполнения всех пунктов задания.

2.3. Варианты заданий

2.3.1. Конструкторы, методы и свойства класса

Создать класс «**Функция**», описывающий объекты - функции из семейства функций заданного вида (вид функции задан в таблице 2.1). Класс должен содержать следующие элементы:

- Закрытые поля для хранения значения параметров функции a и b.
- Свойства для доступа к параметрам функции.
- Метод для вычисления значения функции (входной параметр – значение аргумента x).
- Метод для табулирования функции (входные параметры - $x_{нач}$, $x_{кон}$, шаг Δx). При $x_{нач} <= x_{кон}$ шаг прибавлять, при $x_{нач} > x_{кон}$ шаг вычитать (использовать тернарную операцию **? :**).
- Метод **ToString()**, результатом которого является строковое представление функции.
- Статический метод для ввода начального значения аргумента, конечного значения аргумента и шага изменения аргумента функции (выходные параметры - $x_{нач}$, $x_{кон}$, шаг Δx) Ввод продолжать до тех пор пока не будет введен шаг $\Delta x >= 0$.

Создать класс «**Таблица**», описывающий объекты-таблицы из двух столбцов, содержащий следующие элементы:

- Закрытые поля: заголовок таблицы, заголовки столбцов, ширина первого столбца, ширина второго столбца.
- Конструктор с параметрами.
- Метод для вывода шапки таблицы.
- Метод для вывода строки таблицы (входные параметры – числовые значения, которые выводятся в строке таблицы).
- Метод для вывода низа таблицы.

Разработать программу, которая выполняет следующие действия:

– Создает два объекта класса «**Функция**» (параметры функций вводятся с клавиатуры).
– Для каждого объекта вычисляет значения для трех различных значений аргумента.
– Для каждого объекта–функции выполняет табулирование для ряда значений аргумента. Результат выводить в виде таблицы, в заголовке таблицы приводить вид функции.

Таблица 2.1.

Вариант	Вид функции ***f(x)***
1	$\frac{1+\cos^2(x+a)}{\lvert x^3-2b^2\rvert}$
2	$\frac{\ln^2\lvert b\rvert}{\sqrt[3]{\lvert x\rvert+\lvert a\rvert}}$
3	$\frac{a^3}{x+a^3\cos^2 b}$
4	$\sqrt{x+\sqrt[4]{\lvert a\rvert}}+\cos^2 b$
5	$\frac{\sqrt[3]{e^{\sin x}}\cdot\cos a}{b^2+1}$
6	$a(tg\,b-e^{-(x+3)})$
7	$\lvert x-a\rvert(\sin^2 b+tgx)$
8	$\sqrt{a+\sqrt[3]{x}}-1+2b$

Вариант	Вид функции ***f(x)***
9	$x(tg\,a+\cos bx)$
10	$e^{\lvert x-b\rvert}\left(tg^2a+1\right)^x$
11	$\cos^2 a+tg\ 2x+\lvert b\rvert$
12	$5tg\ a-4x^2+\lvert xb\rvert$
13	$(a-x)\frac{a-\ln b}{1+(b-x)^2}$
14	$a^x+\sqrt{\lvert x\rvert+\lvert b\rvert}$
15	$\frac{\lg\left(\sqrt{x}+\sqrt{a}+2\right)}{\lvert 2b\rvert}$

2.3.2. Работа с объектами класса

Вариант1

Создать класс «**Прямоугольник**», описывающий объекты – прямоугольники со сторонами, параллельными осям координат. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат левого верхнего и правого нижнего углов прямоугольника.
- Конструктор без параметров для создания прямоугольника с левым верхним углом в начале координат и правым нижним углом в точке
  (1;-1).
- Конструктор с параметрами для создания прямоугольника с произвольными координатами углов. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойство для определения площади прямоугольника.
- Метод, результатом которого является **true**, если прямоугольник квадрат, и **false** в противном случае.
- Метод для перемещения прямоугольника по вертикали вверх или по горизонтали вправо (в зависимости от значения соответствующего параметра) на заданную величину.
- Метод для поворота прямоугольника вокруг левого верхнего угла на 90^0 против часовой стрелки.

- Статический метод для проверки, располагается ли один прямоугольник внутри другого (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Прямоугольник**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о прямоугольниках в виде:

№ п/п	Левый верхний угол	Правый нижний	Площадь	Является ли квадратом
1	(0;0)	(1;-1)	1	квадрат
2	(0;5)	(4;3)	8	

– Определяет, располагается ли какой-нибудь прямоугольник внутри другого;
– Осуществляет перемещение или поворот (по выбору пользователя) для второго прямоугольника и выводит новую информацию о нем.

Вариант 2

Создать класс «**Окружность**», описывающий объекты – окружности на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат центра окружности и радиуса.
- Конструктор без параметров для создания окружности с центром в начале координат и единичным радиусом.
- Конструктор с параметрами для создания произвольной окружности. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойства для определения длины окружности.
- Метод, результатом которого является **true**, если окружность целиком лежит в одной координатной четверти, и **false** в противном случае.
- Метод для перемещения окружности по вертикали вниз или по горизонтали влево (в зависимости от значения соответствующего параметра) на заданную величину.
- Метод для увеличения радиуса окружности на заданную величину.

- Статический метод для проверки, пересекаются ли две окружности (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает **три** объекта класса «**Окружность**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию об окружностях в виде:

№ п/п	Центр	Радиус	Длина	Лежит ли в одной координатной плоскости
1	(0;0)	1	6.28	нет
2	(3;5)	2	12.56	да

– Определяет, пересекаются ли какие-нибудь из данных окружностей;
– Осуществляет перемещение или увеличение (по выбору пользователя) для первой окружности и выводит новую информацию о ней.

Вариант 3

Создать класс «**Треугольник**», описывающий объекты – треугольники на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат вершин треугольника.
- Конструктор без параметров для создания треугольника с координатами вершин: (1;-1), (0;0), (1,1).
- Конструктор с параметрами для создания треугольника с произвольными координатами вершин. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойство для определения площади треугольника.
- Метод, результатом которого является **тип** треугольника (равносторонний, равнобедренный, прямоугольный).
- Метод для перемещения треугольника по горизонтали вправо или влево (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для проверки, располагается ли один треугольник внутри другого (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Треугольник**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о треугольниках в таблице:

№ п/п	Координаты вершин	Площадь	Тип
1	(1;-1),(0;0), (1,1)	1	Равнобедренный, прямоугольный

– Определяет, располагается ли какой-нибудь треугольник внутри другого;
– Осуществляет перемещение влево или вправо (по выбору пользователя) для второго треугольника и выводит новую информацию о нем.

Вариант 4

Создать класс «**Отрезок**», описывающий объекты – отрезки на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат концов отрезка.
- Конструктор без параметров для создания отрезка с координатами концов: (0;0), (1,1).
- Конструктор с параметрами для создания произвольного отрезка. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойство для определения длины отрезка.
- Метод, результатом которого является **true**, если отрезок целиком лежит на одной из осей координат, и **false** в противном случае.
- Метод для перемещения окружности по вертикали вниз или вверх (в зависимости от значения соответствующего параметра) на заданную величину.
- Метод для увеличения длины отрезка на заданную величину (один конец отрезка остается на месте, а второй перемещается в соответствующем направлении на заданную величину).
- Статический метод для проверки, пересекаются ли два отрезка (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает **три** объекта класса «**Отрезок**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию об отрезках в виде:

№ п/п	Координаты границ	Длина	Лежит ли на оси координат
1	(0;0), (1;1)	1,4142	нет
2	(1;0), (5;0)	4	да

– Определяет, пересекаются ли какие-нибудь из данных отрезков;
– Осуществляет перемещение или увеличение (по выбору пользователя) для первого отрезка и выводит новую информацию о нем.

Вариант 5

Создать класс «**Параллелограмм**», описывающий объекты – параллелограммы, одна из пар сторон которых параллельна оси абсцисс. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат левого верхнего, левого нижнего углов параллелограмма и длины сторон, параллельных оси абсцисс.
- Конструктор без параметров для создания параллелограмма с левым верхним углом в начале координат, левым нижним углом в точке (1;-1) и длиной сторон, параллельных оси абсцисс, равной 2.
- Конструктор с параметрами для создания параллелограмма с произвольными параметрами. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойство для определения периметра параллелограмма.
- Свойство для определения длины другой пары параллельных сторон.
- Метод, результатом которого является **true**, если параллелограмм является прямоугольником, и **false** в противном случае.
- Метод для перемещения параллелограмма по вертикали вверх или по горизонтали вправо (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для проверки, располагается ли один параллелограмм внутри другого (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Параллелограмм**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о параллелограммах в виде:

№ п/п	Левый верхний угол	Длина одной пары сторон	Длина второй пары сторон	Периметр	Является ли прямоугольником
1	(0;0)	2	1.4142	6.8284	нет
…					

– Определяет, располагается ли какой-нибудь параллелограмм внутри другого;
– Осуществляет перемещение вверх или вправо (по выбору пользователя) для третьего параллелограмма и выводит новую информацию о нем.

Вариант 6

Создать класс «**Дробь**», описывающий множество объектов – дробей вида $\frac{m}{n}$, где *m* и *n* – целые числа, $n \neq 0$. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения числителя и знаменателя дроби.
- Конструктор без параметров для создания дроби $\frac{1}{2}$.
- Конструктор с параметрами для создания произвольной дроби. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для записи).
- Свойство для представления в виде десятичной дроби.
- Статический метод для сложения двух дробей (входные параметры – объекты класса, результат – тоже объект класса).
- Статический метод для вычитания двух дробей (входные параметры – объекты класса, результат – тоже объект класса).
- Статический метод для умножения двух дробей (входные параметры – объекты класса, результат – объект класса**).**
- Метод **ToString()**, результатом которого является строковое представление дроби, например, «1/2».

Разработать программу, которая выполняет следующие действия:
– Создает **четыре** объекта класса «**Дробь**» (один с помощью конструктора без параметров и три произвольных);
– Выводит информацию о дробях в виде:

№ п/п	Дробь	Десятичная дробь
1	1/2	0.5
2	5/2	2.5

– Для третьей дроби вычисляет сумму, разность, произведение с каждой из трех остальных. Результат выводит в виде:

Дробь	2/5+n	2/5-n	2/5*n
3/2	19/10	-11/10	6/10
1/2	9/10	-1/10	2/10
…			

– Осуществляет увеличение числителя или знаменателя (по выбору пользователя) для первой дроби и выводит новую информацию о нем.

Вариант 7

Создать класс «**Квадратное уравнение**», описывающий объекты – уравнения вида $ax^2 + bx + c = 0$. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения коэффициентов уравнения.
- Конструктор без параметров для создания уравнения $x^2 = 1$.
- Конструктор с параметрами для создания уравнения с произвольными коэффициентами. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для записи).
- Свойство для определения дискриминанта.
- Метод **ToString()**, результатом которого является строковое представление уравнения, например, «3x^2+2x-6=0».
- Метод, который вычисляет корни уравнения.
- Статический метод для проверки, совпадают ли корни уравнений (входные параметры – объекты класса, результат - **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Квадратное уравнение**» (один с помощью конструктора без параметров и два произвольных);

– Выводит информацию об уравнениях в таблице:

уравнение	дискриминант	корни
1X^2+0x-1=0	4	1, -1
…		

– Определяет, совпадают ли корни второго и третьего уравнений, первого и третьего уравнений;
– Осуществляет изменение коэффициентов выбранного пользователем уравнения и вычисляет корни измененного уравнения.

Вариант 8

Создать класс «**Определенный интеграл**», описывающий объекты – интегралы вида $\int_a^b \frac{y\sin^2 x - zx}{2.5} dx$. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения пределов интегрирования и параметров y и z подынтегральной функции.
- Конструктор без параметров для создания объекта $\int_0^1 \frac{\sin^2 x - x}{2.5} dx$.
- Конструктор с параметрами для создания интеграла с произвольными параметрами. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса.
- Свойство для определения значения интеграла по формуле Ньютона-Лейбница.
- Метод для определения значения интеграла методом прямоугольников с заданной точностью.
- Закрытый метод для вычисления значения подынтегральной функции.
- Метод для определения значения интеграла методом Симпсона с заданной точностью**.**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Определенный интеграл**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о созданных объектах в таблице:

Подынтегральная функция	Пределы интегрирования	Значение интеграла по ф-ле Ньютона-Лейбница	Значение интеграла методом Симпсона	Значение интеграла методом прямоугольников
(1sin(x)^2-1x)/2.5	0, 1	2,35	2,357	2,345
…				

Вариант 9

Создать класс «**Правильный многоугольник**», описывающий объекты – правильные n-угольники на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат центра описанной окружности и одной из вершин многоугольника, а также количества вершин.
- Конструктор без параметров для создания квадрата с центром описанной окружности в начале координат и вершиной в точке (0; 1).
- Конструктор с параметрами для создания произвольного правильного многоугольника. Предусмотреть проверку на корректность введенных данных.
- Свойство для определения длины стороны.
- Свойства для определения площади и периметра многоугольника.
- Свойство для определения радиуса вписанной окружности.
- Метод, результатом которого является **тип** многоугольника (треугольник, квадрат, 5-угольник, 6-угольник и т.д.).
- Метод для перемещения многоугольника по вертикали вниз или вверх (в зависимости от значения соответствующего параметра) на заданную величину.

Разработать программу, которая выполняет следующие действия:

– Создает **три** объекта класса «**Правильный многоугольник**» (один с помощью конструктора без параметров и два произвольных);

– Выводит информацию о многоугольниках в виде:

№ п/п	Тип	Длина стороны	Площадь	Периметр	Радиус вписанной окружности
1	квадрат	1	1	4	0,8
2	6-угольник	2	12.56	12	1,8

– Определяет, пересекаются ли какие-нибудь из данных окружностей;

– Осуществляет перемещение вверх или вниз (по выбору пользователя) для выбранного многоугольника и выводит новую информацию о нем.

Вариант 10

Создать класс «**Треугольник**», описывающий объекты – треугольники на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат вершин треугольника.
- Конструктор без параметров для ввода координат вершин с клавиатуры при создании объекта. Предусмотреть проверку на корректность введенных данных.
- Конструктор с параметрами для создания треугольника с произвольными координатами вершин. Предусмотреть проверку на корректность введенных данных.
- Свойства для определения длин сторон (только для чтения).
- Свойство для определения углов треугольника.
- Метод для перемещения треугольника по горизонтали вправо или влево (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для проверки, пересекаются ли два треугольника (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

- Создает три объекта класса «**Треугольник**» (один с помощью конструктора без параметров и два с помощью конструктора с параметрами);
- Выводит информацию о прямоугольниках в таблице:

№ п/п	Координаты вершин	Длины сторон	Углы (рад)
1	(0;-1),(0;0), (1,0)	1, 1, 1.4	1.75, 0.87, 0.87

- Определяет, какие треугольники пересекаются;
- Осуществляет перемещение влево или вправо (по выбору пользователя) для первого треугольника и выводит новую информацию о нем.

Вариант 11

Создать класс «**Прямоугольник**», описывающий объекты – прямоугольники. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат вершин прямоугольника.

- Конструктор без параметров для создания прямоугольника с координатами вершин (0;0), (0;1), (1;1), (1;0).
- Конструктор с параметрами для создания прямоугольника с произвольными координатами углов. Предусмотреть проверку на корректность введенных данных.
- Свойства для определения длин сторон (только для чтения).
- Свойство для определения площади прямоугольника.
- Свойство для определения периметра прямоугольника.
- Метод, результатом которого является **true**, если стороны прямоугольника параллельны осям координат, и **false** в противном случае.
- Метод для перемещения прямоугольника по вертикали вверх или по горизонтали вправо (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для проверки, располагается ли один прямоугольник внутри другого (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, выполняющую следующие действия:

– Создает три объекта класса «**Прямоугольник**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о прямоугольниках в виде:

№ п/п	Длины сторон	Периметр	Площадь	Дополнительно
1	1, 1	4	1	Стороны параллельны осям
2	…	…	…	

– Определяет располагается ли какой-нибудь прямоугольник внутри другого;
– Осуществляет перемещение для выбранного пользователем прямоугольника и выводит новые координаты вершин.

Вариант 12

Создать класс «**Окружность**», описывающий объекты – окружности на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат центра окружности и радиуса.

- Конструктор без параметров для создания окружности с центром в начале координат и радиусом, который вводится с клавиатуры. Предусмотреть проверку на корректность радиуса.
- Конструктор с параметрами для создания произвольной окружности. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса.
- Свойство для определения площади круга, ограниченного окружностью.
- Метод, результатом которого является радиус вписанного в окружность правильного n-угольника.
- Метод для перемещения окружности по горизонтали влево или вправо (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для проверки, размещаются ли две окружности в одних и тех же координатных четвертях (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, которая выполняет следующие действия:

– Создает **три** объекта класса «**Окружность**» (один с помощью конструктора без параметров и два с помощью конструктора с параметрами);

– Выводит информацию об окружностях в виде:

№ п/п	Центр	Радиус	Площадь круга	Радиус вписанного многоугольника
1	(0;0)	1	6.28	0.8
2	(3;5)	2	12.56	1.9

– Определяет, размещаются ли какие-нибудь две из данных окружностей в одних и тех же координатных четвертях;

– Осуществляет перемещение влево или вправо (по выбору пользователя) для заданной окружности и выводит новую информацию о ней.

Вариант 13

Создать класс «**Определенный интеграл**», описывающий объекты – интегралы вида $\int_a^b y\cos 2x - zx^2 dx$. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения пределов интегрирования и параметров y и z подынтегральной функции.

- Конструктор без параметров для создания объекта $\int_0^1 \cos 2x - x^2 dx$.
- Конструктор с параметрами для создания интеграла с произвольными параметрами. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса.
- Свойство для определения значения интеграла по формуле Симпсона с точностью 0.001.
- Метод для определения значения интеграла методом правых прямоугольников с заданной точностью.
- Закрытый метод для вычисления значения подынтегральной функции.
- Метод для определения значения интеграла методом трапеций с заданной точностью**.**

Разработать программу, выполняющую следующие действия:

– Создает три объекта класса «**Определенный интеграл**» (один с помощью конструктора без параметров и два произвольных);
– Выводит информацию о созданных объектах в таблице:

Подынтегральная функция	Пределы интегрирования	Значение интеграла методом трапеций	Значение интеграла методом Симпсона	Значение интеграла методом прямоугольников
(1cos(2x)-1x^2)	0, 1	2,35	2,357	2,345
…				

Вариант 14

Создать класс «**Параллелограмм**», описывающий объекты – параллелограммы. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат трех вершин.
- Конструктор без параметров для создания параллелограмма с вводом координат вершин с клавиатуры. Предусмотреть проверку на корректность введенных данных.
- Конструктор с параметрами для создания параллелограмма с произвольными параметрами. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Метод для определения координат четвертой вершины параллелограмма.

- Свойства для определения длин сторон параллелограмма.
- Свойство для определения площади параллелограмма.
- Метод для перемещения параллелограмма по вертикали вниз или по горизонтали влево (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для сравнения периметров двух параллелограммов (входные параметры – объекты класса, результат – параллелограмм с большим периметром**).**

Разработать программу, которая выполняет следующие действия:

– Создает три объекта класса «**Параллелограмм**» (один с помощью конструктора без параметров и два с помощью конструктора с параметрами);
– Выводит информацию о параллелограммах в виде:

№ п/п	Координаты вершин	Длина одной пары сторон	Длина второй пары сторон	Периметр
1	(0;0) (1;2) (5;2) (4;0)	4	2.21	12.42
…				

– Находит параллелограмм с максимальным периметром ;
– Осуществляет перемещение вниз или влево (по выбору пользователя) для третьего параллелограмма и выводит новую информацию о нем.

Вариант 15

Создать класс «**Трапеция**», описывающий объекты – трапеции на координатной плоскости. Класс должен содержать указанные ниже элементы.

- Закрытые поля для хранения координат вершин трапеции.
- Конструктор без параметров для создания трапеции с координатами вершин: (0;0), (1;1), (2;1), (2;0).
- Конструктор с параметрами для создания трапеции с произвольными координатами вершин. Предусмотреть проверку на корректность введенных данных.
- Свойства для доступа к полям класса (только для чтения).
- Свойства для вычисления длин сторон трапеции.
- Свойство для определения площади трапеции.

- Метод, определяющий в каких четвертях координатной плоскости расположена трапеция (результат – строка).
- Метод для перемещения трапеции по горизонтали вправо или влево (в зависимости от значения соответствующего параметра) на заданную величину.
- Статический метод для сравнения периметров двух трапеций (входные параметры – объекты класса, результат – трапеция с большим периметром**).**

Разработать программу, которая выполняет следующие действия:

- Создает три объекта класса «**Трапеция**» (один с помощью конструктора без параметров и два произвольных);
- Выводит информацию о трапециях в таблице:

№ п/п	Координаты вершин	Длины сторон	Площадь	Координатная четверть
1	(0;0), (1;1), (2;1), (2;0)	1.3 1 1 2	1	1

- Находит трапецию с максимальным периметром;
- Осуществляет перемещение влево или вправо (по выбору пользователя) для заданной трапеции и выводит новую информацию о ней.

2.4. Контрольные вопросы

1. Понятие класса. Синтаксис объявления класса. Элементы класса.
2. Вывод данных с помощью методов класса **System.Console**. Форматный вывод. Примеры.
3. Ввод данных с помощью методов класса **System.Console**. Примеры.
4. Синтаксис объявления полей и констант класса. Пример объявления статического и нестатического поля. Обращение к полю и константе. Примеры.
5. Синтаксис метода класса. Спецификаторы методов. Пример метода. Обращение к статическим и нестатическим методам.
6. Виды параметров метода класса. Характеристика каждого вида.
7. Конструктор экземпляра класса. Синтаксис конструктора. Примеры.
8. Статический конструктор и деструктор класса. Синтаксис. Примеры.
9. Свойства класса. Синтаксис. Обращение к свойству. Примеры.

3. Обработка одномерных массивов

Цель: *Получить навыки работы с одномерными массивами, освоить создание и использование одномерных индексаторов класса, ознакомиться с возможностями использования свойств и методов класса Array, научиться применять оператор try для обработки исключений.*

3.1. Краткие теоретические сведения

Исключения перехватываются и обрабатываются оператором **try.**

try
{*контролируемый блок*}
catch (*тип1* [*имя1*]) { *обработчик исключения1* }
catch (*тип2* [*имя2*]) { *обработчик исключения2* }
. . .
catch { *обработчик исключения* }
finally {*блок завершения*}

Исключение можно сгенерировать вручную, используя инструкцию **throw.**

Формат ее записан таков:

throw [*параметр*];

Параметр - это объект класса исключений, производного от класса Exception. Например:

```
double x;
if (x == 0)  throw   new   DivideByZeroException();
```

Одномерный массив можно описать одним из описанных ниже способов.

***тип*[ ] *имя_массива*;**

Например,

double[ ] y, z;

В этом случае память под элементы массивов не выделена.

***тип*[ ] *имя_массива* = new *тип*[*размерность*];**

Здесь размерность - выражение, тип которого имеет неявное преобразование к int, long, ulong, uint.

Например,

int[] a = new int[20], b= new int[100];

В этом случае в памяти создаются массивы из 20 и 100 элементов соответственно, всем элементам присваивается значение 0.

***тип*[ ] *имя_массива* = new *тип*[ ] {*список_инициализаторов*};**

Например,

int[] x = new int[] {2, -5, 0, 9};

Обращение к элементу массива:

***имя массива* [*индекс*]**

Например, x[3], MyArray[10].

Для просмотра всех элементов из некоторой группы данных: массива, списка и др. существует удобный оператор цикла **foreach.**

Синтаксис:

foreach (*тип имя_переменной* in *имя_массива*)
***тело цикла*;**

В приведенном ниже примере показан ввод массива с клавиатуры в консольном приложении.

```
Console.WriteLine("Введите количество элементов");
int n=Convert.ToInt32(Console.ReadLine());
double[ ] x = new double[n];
for (int  i = 0; i < n; ++i)
{
   Console.Write("x[" + i + "]=");
   x[i] = Convert.ToDouble(Console.ReadLine());
}
```

Свойства класса **System.Array** приведены в таблице 3.1.

Таблица 3.1

Свойство	Описание
Length	Количество элементов массива
Rank	Количество размерностей массива

Статические методы класса **System.**Array приведены в таблице 3.2.

Таблица 3.2.

Метод	Описание
Clear(x, j,n)	Присваивает **n** элементам массива **x**, начиная с **j**-го, значения по умолчанию. Например: **Array.Clear(x, 1,2);**
BinarySearch(x, xx)	Ищет в отсортированном массиве **x** номер элемента со значением **xx.** Например, **Array.BinarySearch(a, 9)**
Sort(x)	Упорядочивает массив **x** в порядке возрастания значений элементов
Sort(x, j, n)	Упорядочивает часть массива x из n элементов, начиная с j-го
Reverse(x)	Изменяет порядок следования элементов массива x на обратный.
Reverse(x, j, n)	Изменяет порядок следования n элементов массива x на обратный, начиная с j-го элемента.
Copy(x,z,n)	Копирует n элементов массива x в массив z (n должно быть не больше размерности z и x)
Copy(x,j,z,k,n)	Копирует n элементов массива x, начиная с j-го, в массив z с k-й позиции
IndexOf(x, xx)	Ищет в массиве x номер первого элемента со значением xx.
LastIndexOf(x, xx)	Ищет в массиве x номер последнего элемента со значением xx.

3.2. Варианты заданий

3.2.1. Создание массива объектов

Вариант 1

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о студентах в массив объектов класса Student.

– Вывод списка всех студентов с результатами сдачи сессии.

– Определение количества двоечников в массиве.

Класс Student должен содержать закрытые поля: фамилия, номер группы, успеваемость (массив), индексатор для доступа к элементам массива оценок и все необходимые для решения задачи свойства и методы.

Вариант 2

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о спортсменах в массив объектов класса Sportsmen.

– Вывод списка всех спортсменов с указанием результатов соревнований каждого спортсмена.

– Вывод списка спортсменов, результат которых был меньше заданного не менее одного раза за соревнования (если таких спортсменов нет, вывести соответствующее сообщение).

Класс Sportsmen должен содержать закрытые поля: фамилия, год рождения, результаты соревнований (массив), индексатор для доступа к элементам массива результатов и все необходимые для решения задачи свойства и методы.

Вариант 3

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о сотрудниках предприятия в массив объектов класса Sotrudnik.

– Вывод списка всех сотрудников предприятия с указанием отдела.

– Определение количества сотрудников в заданном отделе, зарплата которых за второй месяц больше средней зарплаты по всему предприятию.

Класс Sotrudnik должен содержать закрытые поля: фамилия, наименование отдела, зарплата за последние n месяцев (массив), индексатор

для доступа к элементам массива зарплат и все необходимые для решения задачи свойства и методы.

Вариант 4

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о преподавателях в массив объектов класса Prepodavatel.

– Вывод списка всех преподавателей университета с указанием кафедры и годовой нагрузки.

– Определение минимальной общей нагрузки за весь учебный год по заданной кафедре.

Класс Prepodavatel должен содержать закрытые поля: фамилия, кафедра, нагрузка в часах 10 месяцев (массив), индексатор для доступа к элементам массива нагрузки и все необходимые для решения задачи свойства и методы.

Вариант 5

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о поездах в массив объектов класса Poezd.

– Вывод информации обо всех поездах с указанием наличия свободных мест в заданном вагоне

– Определение количества свободных мест в поездах, следующих до заданной станции.

Класс Poezd должен содержать закрытые поля: номер поезда, пункт отправления, пункт назначения, количество свободных мест в каждом вагоне (массив), индексатор для доступа к элементам массива свободных мест и все необходимые для решения задачи свойства и методы.

Вариант 6

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о студентах в массив объектов класса Student.

– Вывод списка всех студентов заданной группы с результатами сдачи первого экзамена.

– Определение количества отличников в заданной группе.

Класс Student должен содержать закрытые поля: фамилия, номер группы, успеваемость (массив), индексатор для доступа к элементам массива оценок и все необходимые для решения задачи свойства и методы.

Вариант 7

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о спортсменах в массив объектов класса Sportsmen.

– Вывод списка всех спортсменов с указанием среднего результата соревнований каждого спортсмена.

– Определение количества спортсменов, средний результат которых превысил заданное значение.

Класс Sportsmen должен содержать закрытые поля: фамилия, год рождения, результаты соревнований (массив), индексатор для доступа к элементам массива результатов и все необходимые для решения задачи свойства и методы.

Вариант 8

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о сотрудниках предприятия в массив объектов класса Sotrudnik.

– Вывод списка всех сотрудников предприятия с указанием зарплаты за последние n месяцев.

– Определение средней зарплаты по заданному отделу.

Класс Sotrudnik должен содержать закрытые поля: фамилия, наименование отдела, зарплата за последние n месяцев (массив), индексатор для доступа к элементам массива зарплат и все необходимые для решения задачи свойства и методы.

Вариант 9

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о преподавателях в массив объектов класса Prepodavatel.

– Вывод списка всех преподавателей университета с указанием средней нагрузки каждого преподавателя.

– Вывод списка преподавателей, у которых нагрузка в последнем месяце превышает заданное значение (если таких преподавателей нет, вывести соответствующее сообщение).

Класс Prepodavatel должен содержать закрытые поля: фамилия, кафедра, нагрузка в часах 10 месяцев (массив), индексатор для доступа к элементам массива нагрузки и все необходимые для решения задачи свойства и методы.

Вариант 10

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о поездах в массив объектов класса Poezd.

– Вывод информации о поездах с указанием количества свободных мест в каждом вагоне

– Вывод списка поездов, следующих в заданный город, в которых есть свободные места, с указанием количества вагонов со свободными местами (если таких поездов нет, вывести соответствующее сообщение).

Класс Poezd должен содержать закрытые поля: номер поезда, пункт отправления, пункт назначения, количество свободных мест в каждом вагоне (массив), индексатор для доступа к элементам массива свободных мест и все необходимые для решения задачи свойства и методы.

Вариант 11

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о студентах в массив объектов класса Student.

– Вывод списка всех студентов с указанием среднего балла каждого студента.

– Определение количества студентов, получивших больше двух оценок 10 в массиве.

Класс Student должен содержать закрытые поля: фамилия, номер группы, успеваемость (массив), индексатор для доступа к элементам массива оценок и все необходимые для решения задачи свойства и методы.

Вариант 12

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о спортсменах в массив объектов класса Sportsmen.

– Вывод списка всех спортсменов с указанием года рождения и первого результата соревнований.

– Вывод списка спортсменов моложе заданного возраста если таких спортсменов нет, вывести соответствующее сообщение).

Класс Sportsmen должен содержать закрытые поля: фамилия, год рождения, результаты соревнований (массив), индексатор для доступа к элементам массива результатов и все необходимые для решения задачи свойства и методы.

Вариант 13

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о сотрудниках предприятия в массив объектов класса Sotrudnik.

– Вывод списка всех сотрудников предприятия с указанием средней зарплаты.

– Определение максимальной зарплаты по заданному отделу за последний месяц.

Класс Sotrudnik должен содержать закрытые поля: фамилия, наименование отдела, зарплата за последние n месяцев (массив), индексатор для доступа к элементам массива зарплат и все необходимые для решения задачи свойства и методы.

Вариант 14

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о преподавателях в массив объектов класса Prepodavatel.

– Вывод списка всех преподавателей университета с указанием максимальной месячной нагрузкиВывод списка преподавателей, у которых годовая нагрузка превышает заданное значение меньше чем на 10% (если таких преподавателей нет, вывести соответствующее сообщение).

Класс Prepodavatel должен содержать закрытые поля: фамилия, кафедра, нагрузка в часах 10 месяцев (массив), индексатор для доступа к элементам массива нагрузки и все необходимые для решения задачи свойства и методы.

Вариант 15

Разработать консольное приложение, выполняющее следующие действия:

– Ввод с клавиатуры данных о поездах в массив объектов класса Poezd.

– Вывод информации о поездах с указанием наличия свободных мест в каждом вагоне. Вывод списка поездов, следующих в заданный город, в которых нет свободных мест (если таких поездов нет, вывести соответствующее сообщение).

Класс Poezd должен содержать закрытые поля: номер поезда, пункт отправления, пункт назначения, количество свободных мест в каждом вагоне (массив), индексатор для доступа к элементам массива свободных мест и все необходимые для решения задачи свойства и методы.

3.2. 2. Программирование типовых алгоритмов обработки массивов

Вычислить значение выражения $f = \frac{2\sin a + 3b\cos^3 c}{a+b}$, где a, b и c вычисляются в соответствии со своим вариантом в массивах A, B и C соответственно. Способ вычисления a, b и c указан в таблице 2.1.

Для решения задачи создать класс «Массив», содержащий закрытое поле-массив для хранения данных, методы ввода и вывода элементов массива, свойство для доступа к закрытому полю, а также методы или свойства, выполняющие необходимые действия в соответствии с заданием.

Массивы A и B вводить с клавиатуры, массив C сформировать, скопировав сначала все элементы массива B, расположенные после его левого минимального элемента, а затем элементы массива A, расположенные между правым минимальным элементом и элементом с заданным номером.

При возникновении ошибок должны вырабатываться исключения.

Таблица 3.3

Вариант	Способ вычисления a, b, c
1.	Сумма положительных элементов стоящих после второго максимума
2.	Сумма отрицательных элементов стоящих после 2 го нуля
3.	Произведение положительных элементов после последнего отрицательного
4.	Количество нолей в массиве между минимумом и максимумом
5.	Сумма отрицательных элементов после первого положительного
6.	Количество отрицательных чисел после минимума
7.	Сумма чисел из данного интервала между первым и последним положительным
8.	Количество отрицательных чисел на нечетных местах после первого положительного
9.	Произведение тех чисел, которые равны сумме двух соседних
10.	Количество тех чисел, которые равны сумме элементов, стоящей между максимумом и минимумом.
11.	Произведение отрицательных элементов стоящих после второго максимума
12.	Количество отрицательных элементов стоящих после 1 го нуля
13.	Сумма положительных элементов после последнего отрицательного
14.	Произведение чисел в массиве между минимумом и максимумом
15.	Произведение отрицательных элементов после первого положительного

3.3. Контрольные вопросы

1. Понятие исключения. Стандартные типы исключений. Свойства класса Exception.
2. Обработка исключений оператором **try.** Синтаксис, примеры.
3. Генерация исключений с помощью инструкции **throw**.
4. Способы описания одномерного массива с примерами.
5. Свойства класса **System.Array.** Статические методы **Clear, BinarySearch, Sort** класса **System.Array.** Примеры.
6. Статические методы **Reverse, Copy, IndexOf, LastIndexOf** класса **System.Array.** Примеры.
7. Нестатические методы класса **System.Array.** Примеры.

4. Операции и перегруженные методы класса

Цель: *Освоить средства перегрузки стандартных операций в классе, научиться создавать и использовать перегруженные методы класса, а также методы с переменным числом параметров.*

4.1. Краткие теоретические сведения

Двумерный массив можно описать одним из следующих способов.

***тип*[,] *имя_массива*;**

Например,

double[,] y, z;

В этом случае память под элементы массивов не выделена.

***тип*[,] *имя_массива* = new *тип*[*разм_1*, *разм_2*];**

Например,

int[,] a = new int[5,5], b= new int[10,4];

В этом случае в памяти создаются массивы из 25 и 40 элементов соответственно, всем элементам присваивается значение 0.

***тип*[,] *имя_массива* = new *тип*[,] {*список_инициализаторов*};**

В списке инициализаторов значения сгруппированы в фигурных скобках по строкам. Например,

```
int[,] x = new int[,] {{2, -5,  0, 9},
                       {3, 2, -5, 5},
                       {2, 4, 6, -1}};
```

В этом случае размерность массива явно не указана и определяется по количеству элементов в списке инициализаторов.

***тип*[,] *имя_массива* = {*список_инициализаторов*};**

Операция **new** подразумевается.

***тип*[,] *имя_массива* = new *тип*[*разм1*, *разм2*]**
{*список_инициализаторов*};

Например,

```
int[ , ] x = new int[2,2] {{2, -5}, { 0, 9}};
```

Одномерные индексаторы описываются следующим образом:

```
спецификатор тип  this [тип_индекса индекс]
   {
    get  {код аксессора для получения данных}
    set {код аксессора для установки данных}
   }
```

Многомерные индексаторы описываются следующим образом:

```
спецификатор тип  this [тип1 индекс1, тип2 индекс2, …,
                                          типN  индексN ]
   {
    get  {код аксессора для получения данных}
    set  {код аксессора для установки данных}
   }
```

В классе можно определять следующие унарные операции:

+ - ! ~ ++ -- true false

Синтаксис унарной операции:

public static *тип_результата* operator *знак_операции*
(*тип_операнда операнд*)
{тело_операции}

Тип операнда должен совпадать с классом, для которого определена операция.

Операции **true** и **false** должны быть определены в паре. Нельзя перегружать только одну из них. При этом тип результата обычно **bool**.

Формат этих операций:

public static bool operator true (тип_операнда операнд)
{
тело операции с возвратом значения true *или* false.
}

public static bool operator false (тип_операнда операнд)
{
тело операции с возвратом значения true *или* false.
}

В классе можно определять следующие бинарные операции:

+ - * / %

& | ^ << >>

== != > < >= <=

Синтаксис операции такой:
public static *тип_результата* operator *знак_операции*
(*тип_операнда1* операнд1, *тип_операнда2* операнд2)
{тело_операции}

Тип хотя бы одного из операндов должен совпадать с классом, для которого определена операция.

Операции отношения следует перегружать парами. Например, определяя в классе операцию "<", нужно определить операцию ">", и наоборот.

Пары операций отношения:
<= >=
< >
== !=

Операции преобразования типа преобразуют объект некоторого класса в значение другого типа. Фактически, операция преобразования перегружает операцию приведения типов.

Существуют две формы операторов преобразования: явная и неявная.

Явная форма:

public static explicit operator тип_результата
(исходный_тип параметр)
{return значение;}

Операция выполняет преобразование из типа параметра в тип результата. Одним из этих типов должен быть класс, для которого определяется операция.

Неявная форма:

public static implicit operator тип_результата
(исходный_тип параметр)
{return значение;}

Для одной и той же пары типов, участвующих в преобразовании, **нельзя** определить одновременно обе формы операции преобразования.

Чтобы метод мог принимать произвольное число аргументов, нужно в списке параметров использовать параметр-массив неопределенной длины, помеченный ключевым словом **params**.

Этот параметр может быть только один и должен размещаться в списке последним.

Например,

public int min(int x, int y, params int[ ] z) { }

В этот метод могут быть переданы два и более аргументов.

Параметр с ключевым словом **params** может принять любое количество аргументов, даже нулевое.

Например, определим в классе **Program** метод, вычисляющий сумму нескольких чисел, первое из которых целое, а остальные вещественные.

```
class Program
   {public static double sum(int x, params double[] y)
     { double s = x;
        foreach (double yy in y) s = s + yy;
         return s;
     }
static void Main(string[] args)
     {
        Console.WriteLine("3+5=" + sum(3, 5));
        Console.WriteLine("3+5+2.5=" + sum(3, 5,2.5));
        double[] z= {2,4,6};
        Console.WriteLine("3+ сумма элементов z=" + sum(3,z));
```

```
        Console.ReadKey();
    }
```

4.2. Варианты заданий

Вариант 1

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив целых чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,
- перегруженные методы для поиска максимального элемента во всем массиве и для поиска максимального элемента в части массива, ограниченной начальным и конечным значениями индекса, передаваемых в метод в качестве параметров,
- операция скалярного умножения массивов,
- операции < и > (массив больше, если его размерность больше, если длины массивов одинаковы, большим считается массив, в котором больше положительных элементов).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод двух массивов;
- Поиск максимального элемента в первом массиве;
- Поиск максимального элемента в части второго массива с 3 по элемент с заданным номером;
- Вычисление скалярного произведения массивов;
- Сравнение массивов.
- Если первый массив больше, заменить все элементы этого массива, расположенные после максимального, на значение минимального среди отрицательных элементов второго массива.

Вариант 2

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица целых чисел,
- закрытое строковое поле для хранения имени матрицы,
- свойства для определения количества строк и столбцов массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для вычисления произведения отрицательных элементов матрицы с выводом результата (параметр – имя матрицы) и без вывода,
- операция поэлементного сложения матриц одинаковой размерности,
- операции true и false (массив истинный, если в нем есть ненулевые элементы).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц;
- Вычисление и вывод произведения отрицательных элементов в каждом массиве;
- Вычисление суммы массивов с одинаковыми размерностями;
- Если произведение отрицательных элементов первого массива больше заданного числа, а в третьей матрице есть ненулевые элементы, увеличить все отрицательные элементы этого массива на значение минимального среди элементов последней строки третьего массива.

Вариант 3

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив целых чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,

- статический метод с переменным числом параметров для вычисления общей суммы отрицательных элементов в нескольких массивах,
- операции умножения массива на целое число и числа на массив,
- унарная операция - (знаки элементов меняются на противоположные).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление общей суммы отрицательных элементов в массивах 5*A и C;
- Вычисление общей суммы отрицательных элементов в массивах 2*B, -A и C*4;
- Если сумма отрицательных элементов в массиве –A больше суммы отрицательных элементов в массиве A, заменить все отрицательные повторяющиеся элементы этого массива на значение этой суммы.

Вариант 4

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица вещественных чисел,
- свойства для определения количества строк и столбцов массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для вычисления суммы квадратов элементов матрицы, больших заданного числа (параметр – заданное число), и вычисления суммы квадратов элементов матрицы, расположенных после определенного элемента (параметры – номер строки и номер столбца),
- операция скалярного произведения матриц «правильной» размерности (количество столбцов первой матрицы совпадает с количеством строк второй матрицы),
- операции == и !=.

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц A, B и C;
- Вычисление и вывод суммы квадратов элементов матриц A и B, больших 83.6;
- Вычисление и вывод суммы квадратов элементов матриц C и B*A, расположенных после заданного элемента;
- Если A=B=C, заменить нулями все отрицательные элементы последнего столбца в каждой матрице, в противном случае вычислить C*B, если это возможно.

Вариант 5

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив вещественных чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,
- метод с переменным числом параметров для вычисления суммы элементов массива с заданными номерами, если номера не указаны, вычисляется сумма всех элементов,
- операция сложения массивов (второй массив добавляется в конец первого),
- операции true и false (массив является истинным, если не все его элементы равны нулю).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление суммы элементов массива A+B+C;
- Вычисление суммы элементов с номерами 2, 5,6,8, 12 в массивах A и C;
- Вычисление суммы элементов с четными номерами в массиве B;
- Если в массивах A и B есть ненулевые элементы, заменить все нулевые элементы массива C на сумму элементов массива A+B.

Вариант 6

Создать класс «Квадратная матрица», в котором описать следующие элементы:

- закрытое поле – матрица вещественных чисел,
- открытое поле, содержащее имя матрицы,
- свойство для определения размерности матрицы,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- метод с переменным числом параметров для вычисления произведения элементов диагоналей матрицы, параллельных главной (если параметров нет, вычисляется сумма элементов всех диагоналей, параметры – номера диагоналей: 0 – главная диагональ, 1, 2 и т.д. – выше главной, -1, -2 и т.д. – ниже главной),
- операция поэлементного вычитания матриц одинаковой размерности,
- операция явного преобразования объекта класса «Квадратная матрица» к строке с описанием матрицы (результатом должна стать строка символов, содержащая имя матрицы, размерность, значения максимального и минимального элементов матрицы).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц A, B и C;
- Вывод описания этих матриц с использованием операции преобразования типов;
- Вычисление произведения элементов главной и двух соседних диагоналей матриц A и A-B;
- Вычисление произведения элементов всех диагоналей матрицы C;
- Вычисление произведения элементов диагоналей с заданными номерами матрицы B;
- Если A=B (проверить с помощью операции вычитания), увеличить в два раза все положительные элементы главных диагоналей этих матриц.

Вариант 7

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив вещественных чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,
- перегруженные методы для определения количества отрицательных элементов во всем массиве, определения количества элементов расположенных после элемента с заданным номером, а также для определения количества отрицательных элементов, больших заданного числа,
- операция поэлементного умножения массивов (если массивы разной размерности, в результирующий массив оставшиеся элементы большего массива добавляются без изменения),
- операция неявного преобразования целого числа в объект класса «Одномерный массив» (результатом является массив, размерность которого равна преобразуемому числу, заполненный единицами).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление количества отрицательных элементов массива A*C;
- Вычисление количества отрицательных элементов в массивах A и C, расположенных после элемента с заданным номером;
- Если в массиве A*B есть отрицательные элементы, превышающие -5.3, сформировать массив(объект класса «Одномерный массив») из 500 элементов, в котором все элементы с номерами, кратными 10, равны 0, а остальные равны 1.

Вариант 8

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица целых чисел,
- закрытое строковое поле для хранения имени матрицы,
- свойства для определения количества строк и столбцов массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для нахождения минимального элемента среди всех элементов матрицы, для нахождения минимального среди элементов в четных или нечетных строках (параметр равен 1 для нечетных строк, 2 – для четных) и для нахождения минимального среди элементов, не превышающих заданного значения (параметр – заданное число),
- операции поэлементного умножения матрицы на целое число и числа на матрицу,
- операции > и < (матрица больше другой, если в ней больше элементов, в случае одинакового количества элементов большей считается матрица, в которой меньше отрицательных чисел).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц A, B и C;
- Поиск максимальных элементов в каждой матрице;
- Поиск максимального элемента в матрице 5*A, среди элементов не превышающих число 10;
- Если A>B>C, сформировать одномерный массив из элементов матрицы C, больших максимального среди элементов четных строк матрицы A, и меньших максимального среди элементов нечетных строк матрицы B.

Вариант 9

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив вещественных чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,

- методы ввода и вывода массива,
- перегруженные методы для определения произведения всех элементов массива, для определения произведения элементов массива с номерами кратными заданному числу, а также для определения произведения элементов массива, расположенных до элемента с заданным номером,
- операция поэлементного вычитания массивов (если массивы разной размерности, в результирующий массив оставшиеся элементы большего массива добавляются со знаком +, если этот массив является уменьшаемым, и со знаком -, если вычитаемым),
- операции == и **!=** .

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление произведения элементов с четными номерами в массивах A и C, а также вычисление произведения элементов с номерами кратными 3 массива B;
- Нахождение C-A, A-C и A-B-C;
- Если A не равен B, сформировать массив (объект класса «Одномерный массив») из элементов массива A, больших произведения элементов первой половины массива B.

Вариант 10

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица вещественных чисел,
- свойства для определения количества строк и столбцов массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для вычисления количества элементов матрицы, больших заданного числа (параметр – заданное число), и вычисления количества элементов матрицы, больших заданного числа и расположенных в столбцах с номерами кратными заданному целому числу,

- операция скалярного произведения матриц «правильной» размерности (количество столбцов первой матрицы совпадает с количеством строк второй матрицы),
- операция неявного преобразования матрицы в одномерный массив вещественных чисел (результатом должен быть массив, содержащий суммы элементов строк матрицы).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц A, B и C;
- Определение количества элементов матриц A и B, больших 3.15;
- Нахождение A*B и B*C, если это возможно;
- Если количество положительных элементов в четных столбцах матрицы A совпадает с количеством положительных элементов в столбцах с нечетными номерами матрицы B, сформировать массивы из сумм элементов строк матриц A и B, в противном случае сформировать массив из сумм элементов строк матрицы C.

Вариант 11

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив целых чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,
- статический метод с переменным числом параметров для вычисления общего количества положительных элементов в нескольких массивах,
- операция увеличения элементов массива на целое число,
- операции true и false (массив является истинным, если в нем нет нулей).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;

– Вычисление общего количества положительных элементов в массивах 5+A и C+2;
– Вычисление общего количества положительных элементов в массивах 2+B, A и C+4;
– Если в массиве A нет нулей и в нем больше трех положительных элементов, а в массиве B есть нули, заменить все нулевые элементы массива B на значение среднего арифметического элементов массива A.

Вариант 12

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица целых чисел,
- закрытое строковое поле для хранения имени матрицы,
- свойства для определения количества строк и столбцов массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для вычисления среднего арифметического отрицательных элементов матрицы, которые повторяются более n раз, с выводом результата (параметры – n>=0 и имя матрицы) и без вывода,
- операция поэлементного умножения матриц одинаковой размерности,
- операции true и false (массив истинный, если в нем есть ненулевые элементы).

Разработать программу, выполняющую следующие действия:

– Ввод и вывод трех матриц A, B и C;
– Вычисление и вывод среднего арифметического отрицательных элементов матрицы, которые повторяются более 3 раз в каждой матрице;
– Вычисление A*B*C, если это возможно;
– Если среднее арифметическое отрицательных элементов матрицы A больше заданного числа, а в матрице C есть ненулевые элементы, увеличить все ненулевые элементы массива C на значение минимального среднего арифмети-

ческого отрицательных элементов матрицы A*B, которые повторяются более 2 раз.

Вариант 13

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив вещественных чисел,
- открытое поле с именем массива,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,
- метод с переменным числом параметров для вычисления произведения элементов массива с заданными номерами, если номера не указаны, вычисляется сумма всех элементов,
- операция поэлементного деления массивов одинаковой размерности,
- явная операция преобразования объекта класса «Одномерный массив» в строку с описанием массива (результатом является строка, содержащая имя массива, размерность, количество положительных и отрицательных элементов).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление массивов A и A/B (если это возможно);
- Вычисление произведения элементов с номерами 1, 4,6,9, 11 в массивах B и C;
- Вычисление произведения элементов с нечетными номерами в массиве A;
- Если в массивах A и B совпадают последние элементы, заменить все нулевые элементы массива C на произведение первых трех элементов массива A/B;
- Вывести строковое описание всех полученных массивов.

Вариант 14

Создать класс «Матрица», в котором описать следующие элементы:

- закрытое поле – матрица целых чисел,
- свойства для определения количества строк и столбцов матрицы,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода матрицы,
- перегруженные методы для вычисления суммы квадратов положительных элементов матрицы, расположенных ниже минимального среди элементов строк с номерами кратными n (параметр – n>=0), и вычисления суммы квадратов отрицательных элементов матрицы (без параметров),
- операция поэлементного вычитания матриц одинаковой размерности,
- операции ≤ и ≥ (матрица X≤Y, если сумма квадратов положительных элементов матрицы X меньше или равна сумме квадратов положительных элементов матрицы Y).

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех матриц A, B и C;
- Вычисление и вывод суммы квадратов отрицательных элементов каждой матрицы;
- Вычисление A-B и B-A-C, если это возможно;
- Если A≤ B≤ C, заменить все отрицательные элементы матриц A и B на значение суммы квадратов положительных элементов расположенных ниже минимального среди элементов строк с номерами кратными 3 в матрице C.

Вариант 15

Создать класс «Одномерный массив», в котором описать следующие элементы:

- закрытое поле – массив вещественных чисел,
- свойство для определения длины массива,
- индексатор для доступа к элементам поля-массива,
- конструктор с параметрами,
- методы ввода и вывода массива,

- перегруженные методы для определения суммы всех элементов массива, для определения суммы элементов массива с номерами кратными заданному числу, а также для определения произведения элементов массива, расположенных после элемента с заданным номером,
- операция поэлементного сложения массивов (если массивы разной размерности, в результирующий массив добавляются оставшиеся элементы большего массива),
- операции == и **!=** .

Разработать программу, выполняющую следующие действия:

- Ввод и вывод трех массивов A, B и C;
- Вычисление суммы элементов с нечетными номерами в массивах A, B и C+B, а также вычисление суммы элементов с номерами кратными 4 массива A+B;
- Нахождение A+B+C;
- Если A ≠ B, сформировать массив (объект класса «Одномерный массив») из элементов массива C, больших суммы элементов второй половины массива B и первой половины массива A.

4.3. Контрольные вопросы

1. Способы описания двумерного прямоугольного массива с примерами.
2. Методы класса Array для работы с двумерными массивами. Примеры использования.
3. Одномерные индексаторы. Синтаксис объявления индексатора. Примеры описания и использования.
4. Многомерные индексаторы. Синтаксис объявления индексатора. Примеры описания и использования.
5. Унарные операции. Синтаксис описания. Примеры.
6. Бинарные операции. Синтаксис описания. Примеры.
7. Операции преобразования типа в явной форме. Синтаксис описания. Примеры.
8. Операции преобразования типа в неявной форме. Синтаксис описания. Примеры.
9. Методы с переменным количеством аргументов. Описание. Примеры.

5. Наследование и интерфейсы

Цель: *Освоить механизм наследования, получить навыки разработки и реализации интерфейсов, изучить возможности реализации стандартных интерфейсов библиотеки .NET .*

5.1. Краткие теоретические сведения

Общая форма объявления класса, который наследует базовый класс:

class <*имя_произв._кл.*> : <*имя_базового_кл.* >
{ <тело класса>}

Например,

```
class   Chelovek
 {  public  string fam, dat_rog;
    public  double rost, ves;
    public  void   info()
     {Console.WriteLine(fam +"  " + dat_rog+ " " +rost+ " " +ves);
     }
 }
```

Будем использовать описанный класс в качестве базового для класса «Студент»:

```
class   Student : Chelovek
 { public string vuz, gruppa;
   public int[ ] ocenki ;
   public void info_uch()
      { Console.Write(vuz + "  " + gruppa+"  оценки:" );
        foreach (int x in ocenki) Console.Write("  "+x);
        Console.WriteLine();
      }
 }
```

Тогда в методе Main можно так работать с объектом класса:

```
Student   St1 = new Student();
//обращение к унаследованным полям от класса Chelovek:
St1.fam = "Левкович";   St1.dat_rog = "12.07.89";
 St1.rost = 185;        St1.ves = 78;
//обращение к собственным полям класса
```

St1.ocenki = new int[] { 7, 2, 6 };
St1.vuz="ГГТУ им. П.О.Сухого"; St1.gruppa="ИТ-21";
St1.info( ); // от имени объекта st1 вызван унаследованный метод
St1.info_uch( ); // от имени объекта st1 вызван собственный метод

Если в базовом и производном классах не определены конструкторы, то при создании объекта производного класса используются конструкторы по умолчанию, предоставляемые системой программирования.

Базовые и производные классы могут иметь собственные конструкторы.

Конструктор базового класса инициализирует поля объекта, соответствующие базовому классу, а конструктор производного класса — поля объекта, соответствующие производному классу.

Если в конструкторе производного класса явный вызов конструктора базового класса отсутствует, автоматически вызывается конструктор базового класса без параметров (при его отсутствии – конструктор по умолчанию).

Конструктор производного класса с явным вызовом конструктора базового класса определяется следующим образом:

<имя конструктора>(<*список_параметров*>):
base (<*список_аргументов*>)
{ тело конструктора}

Список аргументов содержит аргументы для конструктора базового класса.

Если в базовом классе несколько конструкторов, то будет выбран конструктор, соответствующий количеству и типу аргументов в списке после слова **base.**

Интерфейс – это тип данных, предназначенный для определения характеристик и поведения, присущих классам, реализующим этот интерфейс.

В интерфейсе задается набор методов, свойств и индексаторов, которые должны быть реализованы в производных классах.

[<спецификаторы>] interface <имя > [: предки]
{<тело интерфейса>}

Спецификаторы – это **public** или **internal** (по умолчанию).

Для вложенных в класс интерфейсов можно использовать спецификаторы **new, protected, private**.

Тело интерфейса составляют заголовки методов, шаблоны свойств и индексаторов, события.

Заголовки методов объявляются следующим образом:

<тип_результата> < имя_метода> (<список_параметров>) ;

В интерфейсе методы неявно являются открытыми (**public**), при этом не разрешается **явным образом** указывать спецификатор доступа.

Шаблон свойства представляется следующим образом:

< тип свойства> <имя свойства>

{ get ; set ;}

Свойство может быть только для чтения **{ get ;}** или только для записи **{ set ;}**

Шаблон индексатора (одномерного) имеет вид:

< тип результата> this[<тип индекса> <имя индекса>]

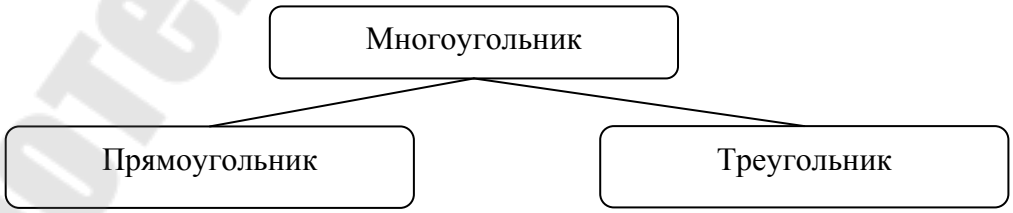
{ get ; set ;}

5.2. Варианты заданий

Замечание: ***Во всех вариантах в классах могут присутствовать элементы, не указанные в задании, указанные элементы обязательны.***

Вариант 1

Создать иерархию классов:



Класс многоугольников должен быть *абстрактным*, содержать следующие элементы: поля (массив, содержащий длины сторон; цвет фигуры); абстрактный метод вычисления площади; метод вывода информации об объекте. Класс многоугольников должен реализовывать интерфейс **`IComparable.`**

Классы прямоугольников и треугольников должны содержать переопределенные методы для вычисления площади.

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит длины сторон прямоугольника или треугольника и цвет фигуры, например: **1 3 3 White;**
- формирует на основании этой информации массив объектов базового класса иерархии;
- выводит на экран всю информацию в виде:

Номер	Вид фигуры	Площадь	Цвет
1	треугольник	15,23	Red

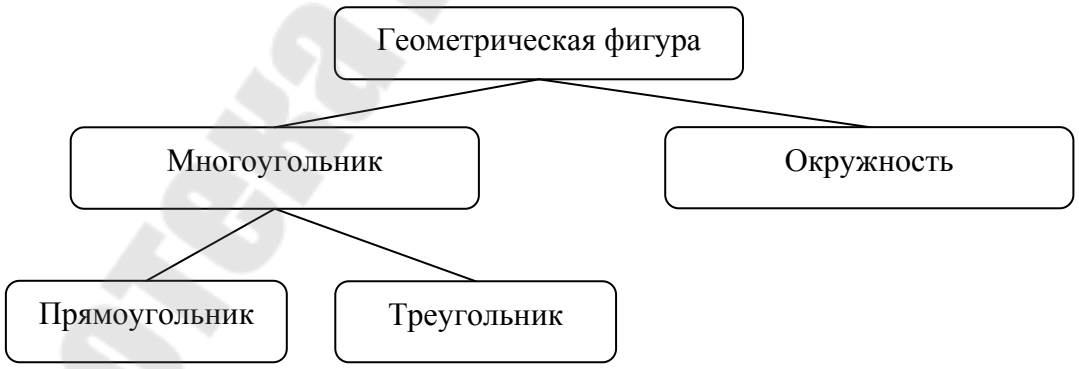
При этом каждая строка выводится тем цветом, который указан в графе цвет.

- сортирует массив в порядке возрастания площадей многоугольников и выводит отсортированный массив;
- вычисляет периметры всех прямоугольных треугольников красного цвета.

Замечание: Для установки цвета символов использовать метод `Enum.Parse(typeof(ConsoleColor),color),` *результат которого преобразуется в объект типа* `ConsoleColor.`

Вариант 2

Создать иерархию классов:



Класс геометрических фигур должен быть *абстрактным*, содержать следующие элементы: поля – тип и цвет фигуры; абстрактный метод вычисления площади; абстрактный метод вывода информации об объекте.

Класс многоугольников должен реализовывать интерфейс `IComparable` и содержать следующие элементы: поле - массив,

содержащий длины сторон; переопределенный метод для вывода информации о фигуре: типа фигуры и площади.

Классы прямоугольников и треугольников должны содержать переопределенные методы для вычисления площади.

Класс окружностей должен содержать поля для хранения координат центра и радиуса, конструктор, переопределенный метод вычисления площади круга, ограниченного окружностью, метод вычисления длины окружности.

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит тип фигуры, длины сторон прямоугольника или треугольника, или координаты центра и радиус окружности, цвет фигуры, например: **многоугольник 1 3 3 White** или **окружность 2 5 3 Red;**
- формирует на основании этой информации массив объектов базового класса иерархии;
- выводит на экран всю информацию в виде:

Многоугольники

Номер	Вид фигуры	Площадь	Цвет
1	треугольник	15,23	Red

Окружности

Номер	Длина окружности	Площадь круга	Цвет
1	13,5	15,23	Red

При этом каждая строка выводится тем цветом, который указан в графе цвет.

- Информацию о многоугольниках размещает в порядке возрастания площадей многоугольников и выводит отсортированный массив;
- вычисляет длины всех окружностей зеленого цвета, целиком расположенных в I четверти координатной плоскости.

Вариант 3

Создать классы квадратов и окружностей, реализующие общий интерфейс «Геометрические фигуры»:

Интерфейс должен определять следующие элементы: свойство возвращающее площадь фигуры, метод вывода информации, индексатор для доступа к параметрам фигуры.

Класс квадратов должен содержать следующие элементы: поле-массив, содержащий координаты вершин, конструктор, реализованные элементы интерфейса, метод вычисления периметра. Класс кругов должен содержать следующие элементы: поля для хранения координат центра и радиуса, цвет фигуры, конструктор, реализованные элементы интерфейса, метод вычисления длины окружности-границы круга.

Дополнительно создать класс, реализующий интерфейс **IComparer.**

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит координаты вершин квадрата или координаты центра, радиус и цвет круга, например:
  1 3 2 White;
- формирует на основании этой информации массив объектов типа «Геометрическая фигура»;
- выводит на экран всю информацию в виде:

Номер	Вид фигуры	Площадь
1	круг	15,23

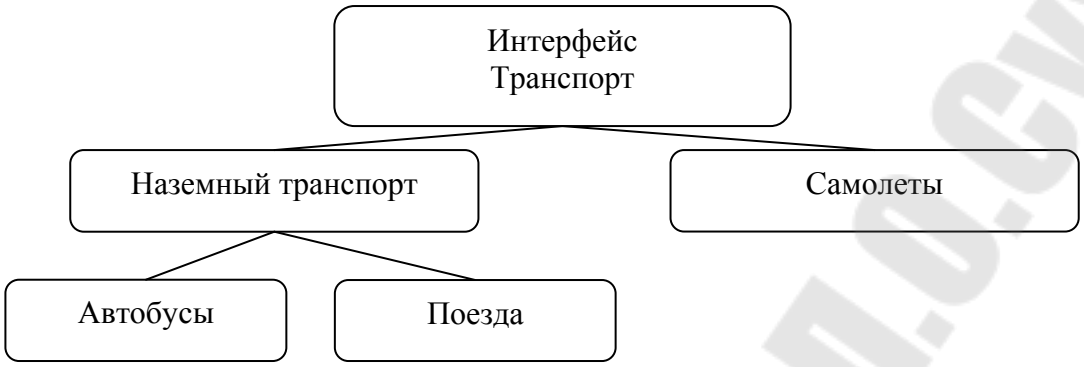
При этом строки с информацией о кругах выводятся тем цветом, который указан в графе цвет.

- сортирует массив в порядке возрастания площадей фигур и выводит отсортированный массив;
- вычисляет периметры всех квадратов, полностью расположенных больше чем в одной четверти координатной плоскости;
- вычисляет длины всех окружностей и выводит полученный результат в порядке убывания длин окружностей.

Замечание: Для установки цвета символов использовать метод `Enum.Parse(typeof(ConsoleColor),color),` *результат которого преобразуется в объект типа* `ConsoleColor.`

Вариант 4

Создать иерархию классов наземного транспорта и класс самолетов, реализующих общий интерфейс «Транспорт»:



Интерфейс должен определять следующие элементы: свойство, возвращающее количество свободных мест; свойства, возвращающие пункт отправления и пункт назначения; свойство, возвращающее вид транспорта; метод вывода информации; индексатор со строковым типом индекса для получения цены на билет.

Класс наземного транспорта должен содержать следующие элементы: поле – номер рейса; поле – массив, содержащий стоимость билета; элементы, реализующие все элементы интерфейса «Транспорт».

Класс автобусов должен содержать поле с количеством свободных мест; переопределять индексатор, в котором индекс может принимать значения **мягкий, жесткий;** переопределять свойство, возвращающее количество свободных мест.

Класс поездов должен содержать поле-массив с количеством свободных мест в каждом вагоне; переопределять индексатор, в котором индекс может принимать значения **люкс**, **купейный, плацкартный, общий;** переопределять свойство, возвращающее количество свободных мест.

Класс самолетов должен содержать поле с количеством свободных мест; поле – номер рейса; поле – массив, содержащий стоимость билета; элементы, реализующие все элементы интерфейса «Транспорт»; переопределять индексатор, в котором индекс может принимать значения **эконом, бизнес, первый**.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит вид транспорта, номер рейса, пункт отправления, пункт назначения, стоимость билетов, например:

самолет 253 Москва Париж 250 280 350;

– формирует на основании этой информации массив объектов типа «Транспорт»;

– вводит всю недостающую информацию;

– Выводит информацию, выделяя красным цветом строки без свободных мест, в виде:

Вид транспорта	Номер рейса	Пункт отправления	Пункт назначения	Количество свободных мест
самолет	256	Гомель	Минск	20

– Выводит цену билета заданного вида на заданный вид транспорта в виде:

Цены билетов на самолет

Номер рейса	Бизнес
256	280

Вариант 5

Создать иерархию классов:

Класс «человек» должен содержать следующие элементы: поле-фамилия, поле-год рождения, поле статус (**студент, преподаватель, бизнесмен и т.д.**), *виртуальный* метод **Svedenija( )** для получения возраста; метод для вывода информации о человеке в виде: **Фамилия статус год рождения сведения.**

Класс студентов должен содержать дополнительное поле-массив с результатами сдачи сессии, переопределенный метод **Svedenija( )** для получения среднего балла за сессию.

Класс преподавателей должен содержать дополнительное поле-массив с нагрузкой по каждой дисциплине; переопределенный метод **Svedenija( )** для получения суммарной годовой нагрузки; индексатор со строковым типом индекса (варианты индекса: **информатика, КИТ, КП и ЯП**).

Дополнительно создать классы, реализующие интерфейс **IComparer,** для получения возможности сортировки по фамилии и по году рождения**.**

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит фамилию, год рождения, статус, для студентов оценки за сессию, для преподавателя нагрузку по дисциплинам, например:
  Иванов 1989 студент 5 6 3 4
  Петров 1978 министр
  Сидоров 1967 преподаватель 54 68 34;
- формирует на основании этой информации массив объектов типа «человек»;
- выводит на экран всю информацию в виде:

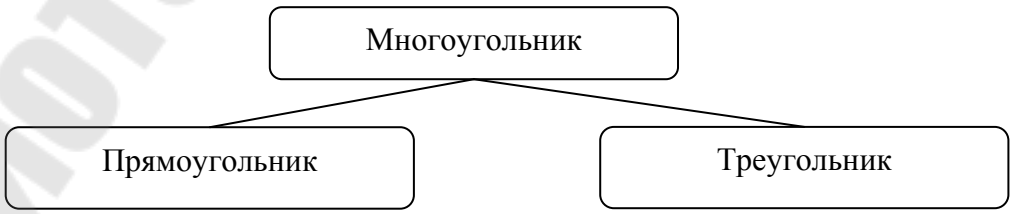
Фамилия	Статус	Год рождения	Дополнительные сведения
Иванов	студент	1989	25 лет
Сидоров	преподаватель	1967	Нагрузка: 156 часов

При этом строки с информацией о людях младше 25 лет выводятся зеленым цветом.

- сортирует массив, располагая фамилии в алфавитном порядке, и выводит отсортированный массив;
- определяет среднюю нагрузку всех преподавателей по заданному предмету;
- выводит информацию о студентах, имеющих больше одной неудовлетворительной оценки, в порядке убывания года рождения.

Вариант 6

Создать иерархию классов:



Класс многоугольников должен быть *абстрактным*, содержать следующие элементы: поля (вид многоугольника; массив, содержащий координаты вершин; цвет фигуры); абстрактные методы вычис-

ления площади и периметра фигуры; метод вывода информации об объекте.

Классы прямоугольников и треугольников должны содержать переопределенные методы для вычисления площади.

Дополнительно создать класс, реализующий интерфейс **IComparer.**

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит вид многоугольника, координаты вершин (для прямоугольника левого верхнего и правого нижнего углов) и цвет фигуры, например:

  треугольник 1 1 2 3 4 1 White;
- формирует на основании этой информации массив объектов базового класса иерархии;
- выводит на экран всю информацию в виде:

Номер	Вид фигуры	Периметр	Площадь	Цвет
1	треугольник	25	15,23	Red

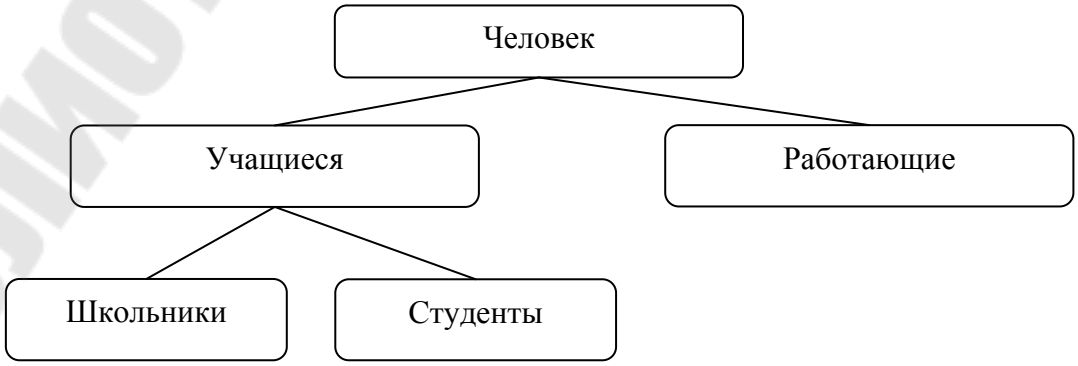
При этом каждая строка выводится тем цветом, который указан в графе цвет.

- сортирует массив в порядке возрастания площадей многоугольников и выводит отсортированный массив;
- меняет цвет всех прямоугольных треугольников, расположенных во II четверти координатной плоскости, на зеленый и выводит измененный массив.

Замечание: Для установки цвета символов использовать метод `Enum.Parse(typeof(ConsoleColor),color),` *результат которого преобразуется в объект типа* `ConsoleColor.`

Вариант 7

Создать иерархию классов:



Класс «человек» должен быть *абстрактным*, содержать следующие элементы: поле-фамилия, поле-год рождения, поле статус (**студент, преподаватель, бизнесмен и т.д.**), абстрактный метод **Svedenija( )** для получения дополнительных сведений о человеке; метод для вывода информации о человеке в виде: **Фамилия статус год рождения сведения.**

Класс учащихся должен реализовывать интерфейс **`IComparable`** и содержать следующие элементы: поле с названием учебного учереждения; поле - массив, содержащий оценки; переопределенный метод **Svedenija( )**, вычисляющий средний балл.

Классы школьников и студентов должны содержать перегруженный метод вывода информации в виде:

Фамилия школа (или ВУЗ) класс(или группа) статус год рождения сведения.

В классе школьников должно быть дополнительное числовое поле с номером класса, а в классе студентов строковое поле с названием группы.

Класс работающих должен содержать поля для хранения места работы, должности, поле-массив с зарплатой за каждый из 12 месяцев; переопределенный метод **Svedenija( )**, возвращающий максимальную зарплату.

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит фамилию, год рождения, статус, для учащихся название учебного заведения, оценки, номер класса или название группы; для работающих место работы, зарплату, например:

Иванов 1989 студент ГГТУ ИТ-21 5 6 3 4
Сидоров 1967 преподаватель ГГТУ 540 680 342 690 790 ;

- формирует на основании этой информации массив объектов типа «человек»;
- выводит на экран всю информацию в виде:

Фамилия	Статус	Возраст	Дополнительные сведения
Иванов	студент	1989	4,5
Сидоров	преподаватель	1967	790

При этом строки с информацией о школьниках старше 12 лет выводятся желтым цветом.

- Выводит информацию об учащихся, располагая фамилии в алфавитном порядке;

– Определяет количество двоечников в заданной школе и выводит информацию о них в виде:

Фамилия школа класс статус год рождения сведения

– выводит информацию о студентах, претендующих на повышенную стипендию по результатам сессии.

Вариант 8

Создать иерархию классов:

Класс геометрических фигур должен быть *абстрактным*, содержать следующие элементы: поля – тип и цвет фигуры; абстрактный метод вычисления площади; метод вывода информации об объекте.

Класс многоугольников должен реализовывать интерфейс **`IComparable`** и содержать следующие элементы: поле - массив, содержащий координаты вершин многоугольника; метод для вычисления периметра.

Классы прямоугольников и треугольников должны содержать переопределенные методы для вычисления площади.

Класс окружностей должен содержать поля для хранения координат центра и радиуса, конструктор, переопределенный метод вычисления площади круга, ограниченного окружностью.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит тип фигуры, координаты вершин прямоугольника или треугольника, или координаты центра и радиус окружности, цвет фигуры, например: **многоугольник 1 3 3 White** или **окружность 2 5 3 Red;**

– формирует на основании этой информации массив объектов базового класса иерархии;
– выводит на экран всю информацию в виде:

Номер	Вид фигуры	Площадь	Цвет
1	треугольник	15,23	Red
2	окружность	25	Green

При этом каждая строка выводится тем цветом, который указан в графе цвет.

– Информацию размещает в порядке возрастания площадей и выводит отсортированный массив;
– вычисляет периметры всех равносторонних треугольников, целиком расположенных во II четверти координатной плоскости.

Вариант 9

Создать классы квадратов и окружностей, реализующие общий интерфейс «Геометрические фигуры»:

Интерфейс должен определять следующие элементы: свойство, возвращающее площадь фигуры, метод вывода информации, индексатор для доступа к параметрам фигуры.

Класс квадратов должен содержать следующие элементы: поле-массив, содержащий координаты вершин, цвет фигуры, конструктор, реализованные элементы интерфейса, метод вычисления периметра. Класс кругов должен содержать следующие элементы: поля для хранения координат центра и радиуса, цвет фигуры, конструктор, реализованные элементы интерфейса.

Дополнительно создать класс, реализующий интерфейс **IComparer.**

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит координаты вершин квадрата или координаты центра, радиус круга; цвет фигуры, например:

1 3 2 White;

– формирует на основании этой информации массив объектов типа «Геометрическая фигура»;
– выводит на экран всю информацию в виде:

Номер	Вид фигуры	Цвет	Площадь
1	круг	**White**	15,23

При этом строки с информацией о кругах выводятся тем цветом, который указан в графе цвет.

– сортирует массив в порядке возрастания площадей фигур и выводит отсортированный массив в виде:

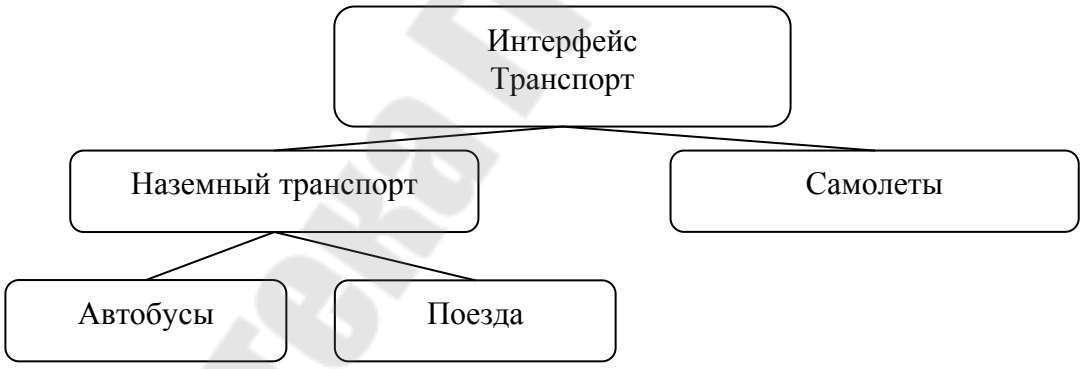
Номер	Вид фигуры	Площадь	Положение
1	круг	15,23	I, II четверть

– вычисляет периметры всех квадратов красного цвета и выводит полученный результат в порядке убывания периметров.

Замечание: Для установки цвета символов использовать метод `Enum.Parse(typeof(ConsoleColor),color),` *результат которого преобразуется в объект типа* `ConsoleColor.`

Вариант 10

Создать иерархию классов наземного транспорта и класс самолетов, реализующих общий интерфейс «Транспорт»:



Интерфейс должен определять следующие элементы: свойство, возвращающее количество свободных мест; свойства, возвращающие пункт отправления и пункт назначения; свойство, возвращающее вид транспорта; метод вывода информации; индексатор со строковым типом индекса для получения цены на билет.

Класс наземного транспорта должен содержать следующие элементы: поле – номер рейса; поле – массив, содержащий стоимость билета; элементы, реализующие все элементы интерфейса «Транспорт».

Класс автобусов должен содержать поле с количеством свободных мест; переопределять индексатор, в котором индекс может принимать значения **мягкий, жесткий;** переопределять свойство, возвращающее количество свободных мест.

Класс поездов должен содержать поле-массив с количеством свободных мест в каждом вагоне; переопределять индексатор, в котором индекс может принимать значения **люкс**, **купейный, плацкартный, общий;** переопределять свойство, возвращающее количество свободных мест.

Класс самолетов должен содержать поле с количеством свободных мест; поле – номер рейса; поле – массив, содержащий стоимость билета; элементы, реализующие все элементы интерфейса «Транспорт»; переопределять индексатор, в котором индекс может принимать значения **эконом, бизнес, первый**.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит вид транспорта, номер рейса, пункт отправления, пункт назначения, стоимость билетов и количество свободных мест, например:

самолет 253 Москва Париж 250 280 350 места: 12
поезд 95 Гомель Москва 286 250 210 160 места: 4 8 9 3 6 12;

– формирует на основании этой информации массив объектов типа «Транспорт»;

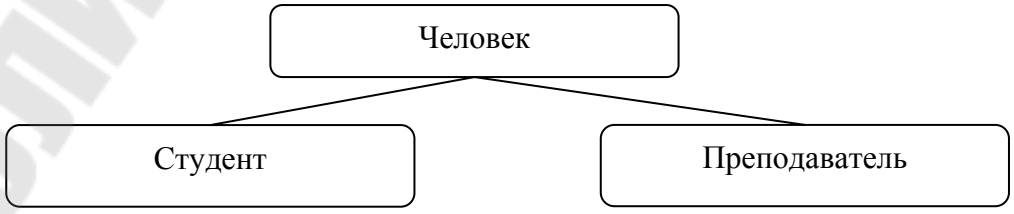
–Выводит информацию, выделяя красным цветом строки без свободных мест, в виде:

Вид транспорта	Номер рейса	Пункт отправления	Пункт назначения	Количество свободных мест
самолет	256	Гомель	Минск	20

– Определяет рейс с минимальной ценой билета заданного вида на заданный вид транспорта.

Вариант 11

Создать иерархию классов:



Класс «человек» должен содержать следующие элементы: поле-фамилия, поле-год рождения, поле статус (**студент, преподаватель, бизнесмен и т.д.**), *виртуальный* метод **Svedenija( )** для получения возраста; метод для вывода информации о человеке в виде: **Фамилия статус год рождения сведения.**

Класс студентов должен содержать дополнительное поле-массив с результатами сдачи сессии, переопределенный метод **Svedenija( )** для получения среднего балла за сессию.

Класс преподавателей должен содержать дополнительное поле-массив с нагрузкой по каждой дисциплине; переопределенный метод **Svedenija( )** для получения суммарной годовой нагрузки; индексатор со строковым типом индекса (варианты индекса: **информатика, КИТ, КП и ЯП**).

Класс «человек» должен реализовывать интерфейс **`IComparable`** для получения возможности сортировки по фамилии**.**

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит фамилию, год рождения, статус, для студентов оценки за сессию, для преподавателя нагрузку по дисциплинам, например:
  Иванов 1989 студент 5 6 3 4
  Петров 1978 бомж
  Сидоров 1967 преподаватель 54 68 34;
- формирует на основании этой информации массив объектов типа «человек»;
- выводит на экран всю информацию в виде:

Фамилия	Статус	Год рождения	Дополнительные сведения
Иванов	студент	1989	25 лет
Сидоров	преподаватель	1967	Нагрузка: 156 часов

При этом строки с информацией о студентах старше 19 лет выводятся красным цветом.

- сортирует массив, располагая фамилии в алфавитном порядке, и выводит отсортированный массив в виде:

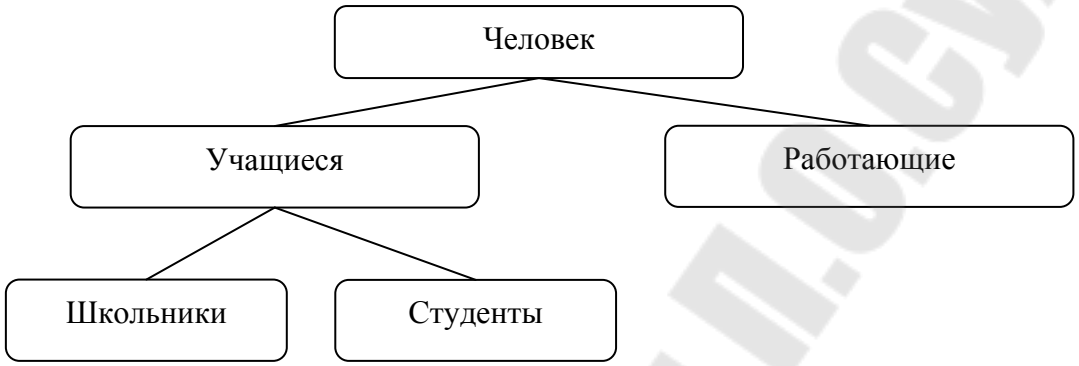
Фамилия	Статус	Возраст
Иванов	студент	18
Сидоров	преподаватель	35

- определяет максимальную нагрузку среди всех преподавателей старше 40 лет по заданному предмету;

– выводит информацию о студентах, имеющих больше одной оценки 9.

Вариант 12

Создать иерархию классов:



Класс «человек» должен быть *абстрактным*, содержать следующие элементы: поле-фамилия, поле-год рождения, поле статус (**студент, преподаватель, бизнесмен и т.д.**), абстрактный метод **Svedenija( )** для получения дополнительных сведений о человеке; метод для вывода информации о человеке в виде: **Фамилия статус год рождения сведения.**

Класс учащихся и содержать следующие элементы: поле с названием учебного учереждения; поле - массив, содержащий оценки; переопределенный метод **Svedenija( )**, вычисляющий средний балл.

Классы школьников и студентов должны содержать перегруженный метод вывода информации в виде:

Фамилия школа (или ВУЗ) класс(или группа) статус год рождения сведения.

В классе школьников должно быть дополнительное числовое поле с номером класса, а в классе студентов строковое поле с названием группы.

Класс работающих должен реализовывать интерфейс **`IComparable`** и содержать поля для хранения места работы, должности, поле-массив с зарплатой за каждый из 12 месяцев; переопределенный метод **Svedenija( )**, возвращающий максимальную зарплату.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит фамилию, год рождения, статус, для учащихся название учебного заведения, оценки,

номер класса или название группы; для работающих место работы, зарплату, например:

Иванов 1989 студент ГГТУ ИТ-21 5 6 3 4

Сидоров 1967 преподаватель ГГТУ 540 680 342 690 790;

- формирует на основании этой информации массив объектов типа «человек»;
- выводит на экран всю информацию в виде:

Фамилия	Статус	Возраст	Дополнительные сведения
Иванов	студент	1989	4,5
Сидоров	преподаватель	1967	790

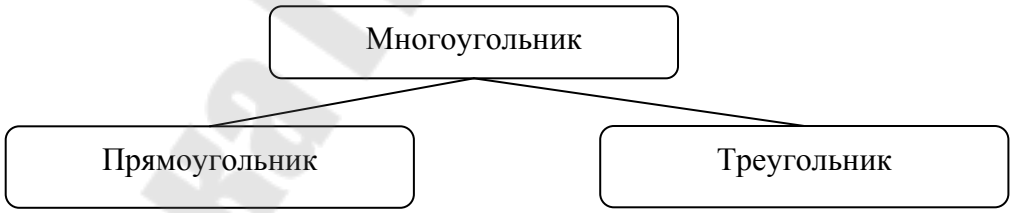
При этом строки с информацией о школьниках-отличниках младше 10 лет выводятся зеленым цветом.

- Выводит информацию о работающих, располагая фамилии в алфавитном порядке;
- Определяет количество двоечников в заданном ВУЗе и выводит информацию о них в виде:

  Фамилия группа сведения
- выводит список школьников, имеющих больше одной двойки.

Вариант 13

Создать иерархию классов:



Класс многоугольников должен быть *абстрактным*, содержать следующие элементы: поля (массив, содержащий координаты вершин; цвет фигуры); абстрактный метод вычисления площади; метод вывода информации об объекте. Класс многоугольников должен реализовывать интерфейс **`IComparable.`**

Классы прямоугольников и треугольников должны содержать переопределенные методы для вычисления площади.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит цвет фигуры и координаты вершин прямоугольника или треугольника, например:

Red 1 3 3 5 2 -2;

– формирует на основании этой информации массив объектов базового класса иерархии;

– выводит на экран всю информацию в виде:

Номер	Вид фигуры	Площадь	Цвет
1	треугольник	15,23	Red

При этом каждая строка выводится тем цветом, который указан в графе цвет.

– сортирует массив в порядке возрастания площадей многоугольников и выводит отсортированный массив;

– Выводит информацию о тех фигурах, которые полностью расположены в одной из четвертей координатной плоскости в виде:

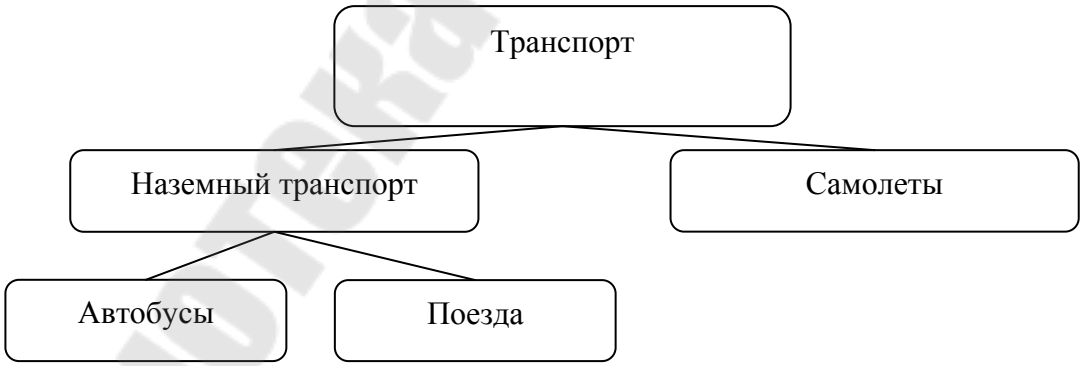
.

Номер	Вид фигуры	Положение
1	треугольник	III

Замечание: Для установки цвета символов использовать метод `Enum.Parse(typeof(ConsoleColor),color),` *результат которого преобразуется в объект типа* `ConsoleColor.`

Вариант 14

Создать иерархию классов:



Класс «транспорт» должен быть абстрактным и содержать следующие элементы: абстрактный метод, возвращающий количество свободных мест; поля - пункт отправления и пункт назначения, вид транспорта; метод вывода информации.

Класс наземного транспорта должен содержать следующие элементы: поле – номер рейса; поле – массив, содержащий стоимость билета.

Класс автобусов должен содержать поле с количеством свободных мест; индексатор, в котором индекс может принимать значения **мягкий, жесткий;** переопределять метод, возвращающее количество свободных мест.

Класс поездов должен содержать поле-массив с количеством свободных мест в каждом вагоне; индексатор, в котором индекс может принимать значения **люкс**, **купейный, плацкартный, общий;** переопределять метод, возвращающий количество свободных мест.

Класс самолетов должен содержать поле с количеством свободных мест; поле – номер рейса; поле – массив, содержащий стоимость билета; индексатор, в котором индекс может принимать значения **эконом, бизнес, первый**.

Разработать программу, которая выполняет следующие действия:

– считывает информацию из текстового файла, каждая строка которого содержит вид транспорта, номер рейса, пункт отправления, пункт назначения, стоимость билетов, например:

самолет 253 Москва Париж 250 280 350;

– формирует на основании этой информации массив объектов типа «Транспорт»;

– вводит всю недостающую информацию;

– Выводит информацию, выделяя красным цветом строки с информацией о самолетах, в виде:

Вид транспорта	Номер рейса	Пункт отправления	Пункт назначения	Количество свободных мест
самолет	256	Гомель	Минск	20

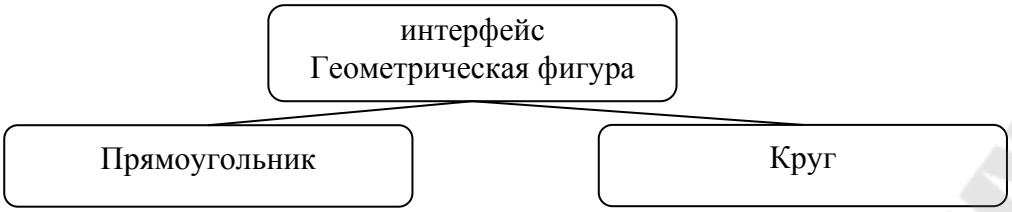
– Выводит цену билета заданного вида на заданный вид транспорта в виде:

Цены билетов на самолет

Номер рейса	Пункт отправления	Пункт назначения	Бизнес
256	Гомель	Минск	280

Вариант 15

Создать классы прямоугольников и кругов, реализующие общий интерфейс «Геометрические фигуры»:



Интерфейс должен определять следующие элементы: свойство, возвращающее площадь фигуры, метод вывода информации, свойства для доступа к параметрам фигуры.

Класс прямоугольников должен содержать следующие элементы: поле-массив, содержащий координаты вершин, конструктор, реализованные элементы интерфейса, метод вычисления периметра. Класс кругов должен содержать следующие элементы: поля для хранения координат центра и радиуса, цвет круга, конструктор, реализованные элементы интерфейса.

Дополнительно создать класс, реализующий интерфейс **IComparer.**

Разработать программу, которая выполняет следующие действия:

- считывает информацию из текстового файла, каждая строка которого содержит координаты вершин прямоугольника или координаты центра, радиус круга; цвет фигуры, например:
  1 3 2 зеленый;
- формирует на основании этой информации массив объектов типа «Геометрическая фигура»;
- выводит на экран всю информацию в виде:

Номер	Вид фигуры	Цвет	Площадь
1	круг	**зеленый**	15,23

- сортирует массив в порядке возрастания площадей фигур и выводит отсортированный массив в виде:

Номер	Вид фигуры	Площадь
1	круг	15,23

- вычисляет периметры всех прямоугольников красного цвета и выводит полученный результат в порядке убывания периметров.

5.3. Контрольные вопросы

1. Понятие наследования. Создание производных классов.
2. Доступ к элементам базового класса

3. Использование конструкторов базового класса
4. Переопределение элементов базового класса. Сокрытие имен.
5. Абстрактные классы. Абстрактные методы. Синтаксис, переопределение. Примеры
6. Виртуальные методы. Описание, переопределение, примеры.
7. Понятие интерфейса. Описание интерфейса. Примеры.
8. Реализация интерфейса в классе. Создание объектов интерфейсного типа. Примеры.
9. Операции is и as. Синтаксис. Примеры.
10. Интерфейс **IComparable.** Его реализация в классе. Примеры.
11. Интерфейс **IComparer.** Его реализация в классе. Примеры.

СОДЕРЖАНИЕ

Романькова Татьяна Леонидовна

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лабораторный практикум по одноименной дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 11.06.14.
Рег. № 87Е.
http://www.gstu.by